18

The Meteorological Anchor Desk System

A Case Study in Building a Web-Based System from Off-the-Shelf Components

with Rob Veltre and Nelson Weiderman

Of course, a component is another system. The nature of a system is such that at almost any granularity it looks the same—it is a system. . . . The goal is to build a system which is necessarily made up of other systems.
— Richard P. Gabriel

In this chapter we examine a system called the Meteorological and Oceanographic (METOC) Anchor Desk System, a network-based collaborative information-gathering and decision-aiding system used by the U.S. military. This case study is an illustration of a system built almost entirely from preexisting large-grained components and of the Evolutionary Development method that promoted this kind of construction.

The METOC Anchor Desk System is a decentralized help desk for meteorological and environmental information. It is designed for crisis situations but also has utility during normal operations. The users of the METOC Anchor Desk (during crisis) are the meteorological officers of the various U.S. military services in

Note: Rob Veltre is a project scientist at the School of Computer Science, Carnegie Mellon University.

Nelson Weiderman is a member of the technical staff at the Software Engineering Institute, Carnegie Mellon University.

Excepted from Bass et al., Software Architecture in Practice, First Edition (ISBN-13: 9780201199307)
Copyright © 1998 Pearson Education, Inc. Do not redistribute.

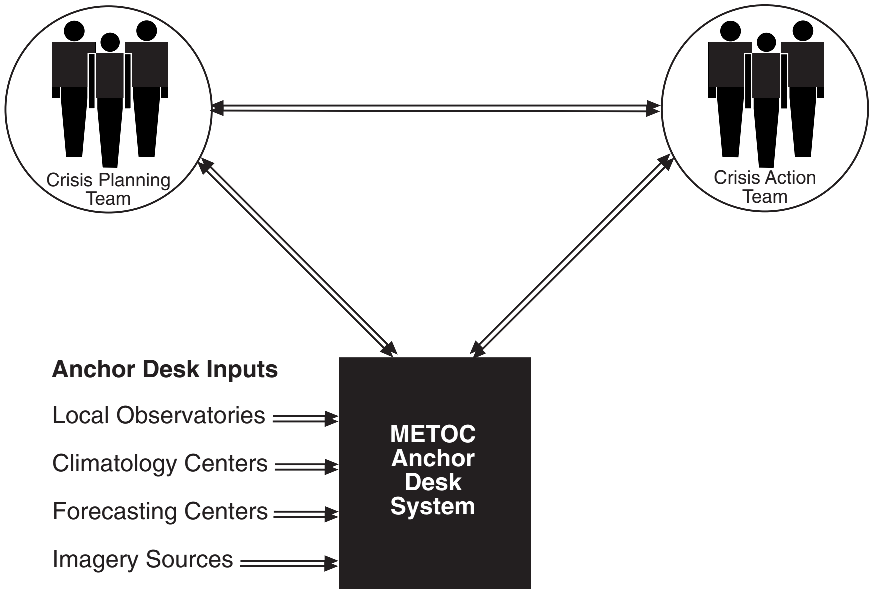

FIGURE 18.1 METOC Anchor Desks concept.

the Pacific Theater Headquarters in Hawaii. They provide support to the crisis planning and action teams that are established by the Pacific Theater Commander in Chief in response to a specific crisis.

Figure 18.1 shows the sources of information for the METOC Anchor Desk and the two different teams involved. While the METOC Anchor Desk is illustrated as a single box in the figure, it is really a globally decentralized system rather than a localized physical entity. The types of crises that are envisioned are both military (during conflict) and civilian (natural disaster relief, drug enforcement, and local law enforcement). During normal operations the same meteorological officers use the METOC Anchor Desk to generate information for their respective commanders.

For example, Figure 18.2 shows a satellite image of a set of dangerous weather systems that developed off the southeast coast of the United States in August of 1996: Hurricane Edouard, Hurricane Fran, and Tropical Storm Gustav. One purpose of the METOC Anchor Desk is to make this kind of information available on a timely basis to crisis response teams.

Excepted from Bass et al., Software Architecture in Practice,
First Edition (ISBN-13: 9780201199307)
Copyright © 1998 Pearson Education, Inc. Do not redistribute.

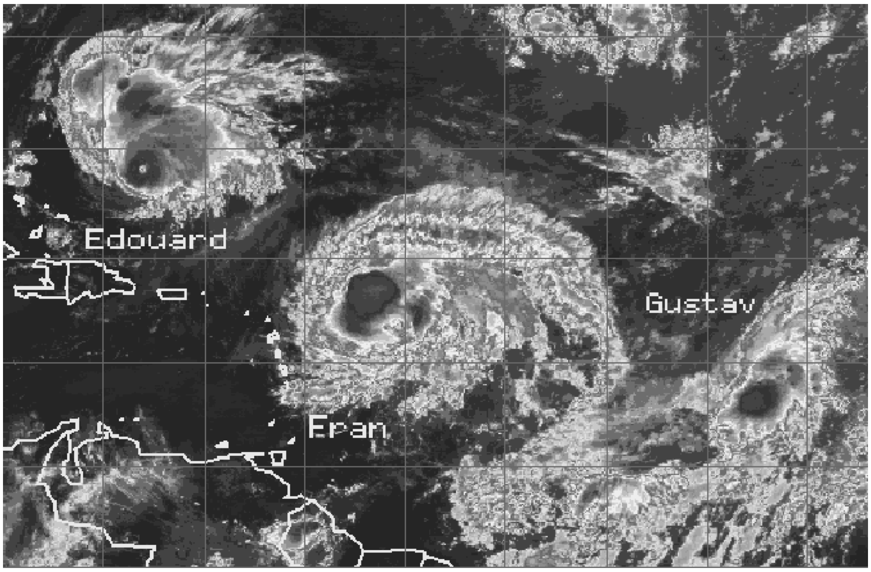

FIGURE 18.2 Weather systems threatening the eastern United States, August 1996. NOAA image. Reproduced with permission.

18.1 Relationship to the Architecture Business Cycle

This case study is situated in the architecture business cycle (ABC) as shown in Figure 18.3. The system was developed using a methodology called Evolutionary Development. This method emphasizes early prototyping and customer feedback. In terms of the ABC, the Evolutionary Development method represents a fine-grained feedback from the system to the developing organization, which occurs while the system is undergoing development instead of after it has been released.

This case study differs from others in the book in that the METOC Anchor Desk System was constructed by an organization primarily interested in the process of software development rather than the products of that development. As such, we will not explore the business issues relevant to the developer.

Excepted from Bass et al., Software Architecture in Practice, First Edition (ISBN-13: 9780201199307)
Copyright © 1998 Pearson Education, Inc. Do not redistribute.

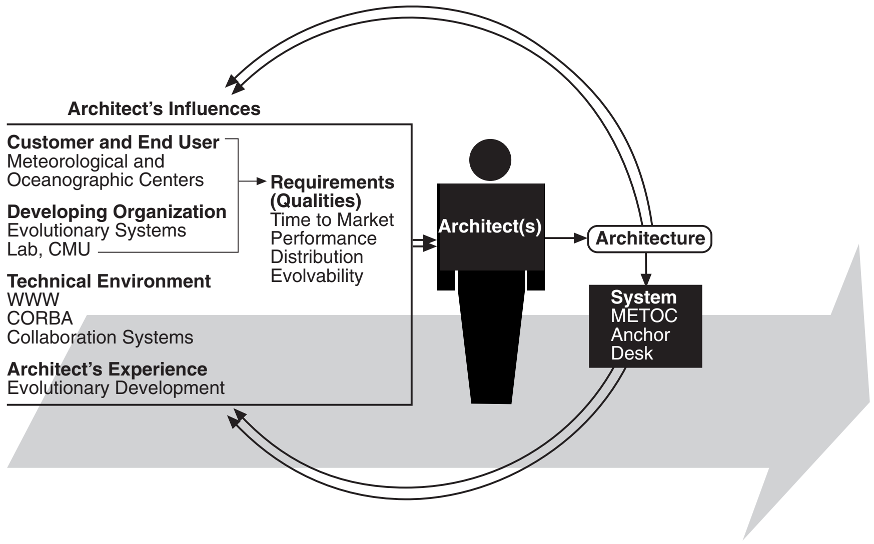

FIGURE 18.3 METOC and ABC.

18.2 Requirements and Qualities

The METOC Anchor Desk is realized as a distributed, virtual organization overlaid on top of existing METOC organizations. These organizations have a long history of operation and have joint procedures that are established and work well. The METOC Anchor Desk System does not attempt to change those procedures. Rather, it attempts to make the existing organization more efficient in executing the procedures. Also, it adds capabilities that improve the timeliness, accuracy, and effectiveness of the decision making under the existing procedures. Because the participants are geographically distributed, collaboration technology such as desktop audio/video, shared whiteboard, and shared applications plays a prominent role in providing improved decision-making abilities.

From a user's point of view, the mission of the METOC Anchor Desk is to do the following:

- Provide environmental decision support to both planners and action team commanders
- Assemble and integrate environmental information
- Assimilate and interpret data for the commanders

The following resources that can be brought to bear on the user's planning problems:

- Robust support for distributed collaborative planning

Excepted from Bass et al., Software Architecture in Practice, First Edition (ISBN-13: 9780201199307)
Copyright © 1998 Pearson Education, Inc. Do not redistribute.

- A hypermedia web of environmental products and crisis management procedures
- Environmental METOC decision aids
- Legacy tools from earlier METOC systems

METOC ANCHOR DESK INPUTS AND OUTPUTS

Typical inputs to the METOC Anchor Desk could include any kind of request for environmental support or tailored meteorological products during any phase of a crisis, from situation development through execution. For example, during situation development, the METOC Anchor Desk may be asked to develop an environmental profile of the situation area, including climatological analysis. Situation analysis can be based on historical data, current satellite photographs, wind strength and direction, wave heights, cloud cover, barometric readings, and so on. The result of the analysis is provided in the form of a forecast of various weather conditions. Such information is useful during the course of action development, evaluation, and execution planning because the provisioning of personnel and the preparation of equipment are directly influenced by the weather considerations.

As another example, the METOC Anchor Desk may be asked to respond rapidly to emergent situations, such as providing quick turnaround surf forecasts for a previously unplanned amphibious landing. The anchor desk may also be asked to put up a classified Web server with environmental products tailored to the operation, thereby providing on-demand access to relevant environmental information.

Currently, requests for support can occur in either of two forms, as follows:

1. Person to person, either in person, by telephone, by video conference, by collaborative planning tools, or through some combination of the above
2. Via the World Wide Web (WWW), either through the use of forms or routine access to environmental product Web pages

TOOLS FOR COLLABORATION AND COMMUNICATION

Anchor desks are intended to enhance, not replace, person-to-person communication. The METOC Anchor Desk System's video-conferencing and collaboration tools permit people who are geographically remote from one another to collaborate about a common task. High-bandwidth networks permit the transmission of live video and audio.

While video conferencing includes video and audio communication, much more than that is required for true collaboration. Visual imagery and hypermedia provide much higher communication bandwidth and cognition than linear text alone. To collaborate remotely with the same effectiveness as one would in the same room, users simultaneously share documents, maps, and diagrams and write on them and modify them in real time in a manner such that all the participants can see the results. For the METOC Anchor Desk System, the shared whiteboard

Excepted from Bass et al., Software Architecture in Practice, First Edition (ISBN-13: 9780201199307)
Copyright © 1998 Pearson Education, Inc. Do not redistribute.

gives all the participants in the conference the ability to do the following with each of the participants seeing the same whiteboard image on a local screen:

- Load a graphic image (maps, satellite image, or forecasts) from a separate file such a GIF or TIFF format image
- Point to items of interest (with a unique cursor for each participant)
- Draw on the image using MacDraw or similar drawing programs
- Switch whiteboard pages
- Scan and digitize a paper image for display

A screen showing several windows of a video conference session is shown in Figure 18.4. In the foreground are two participants in the collaboration and in the background is a weather depiction on a shared whiteboard.

In addition, a file transfer tool (to send text or binary files from one participant to another), a screen capture tool (to capture a user-designated area of the screen), and a “chat” tool (for participants to type messages to each other in the event that the audio fails) are provided. One of the most effective independent tools is called Shared-X. This tool works in workstation environments supporting the X-Windowing System. It gives a participant in a conference the ability to delegate to a remote participant control of a local computer process. This is particularly useful when the remote participant may be more knowledgeable about a product or problem than the local participant.

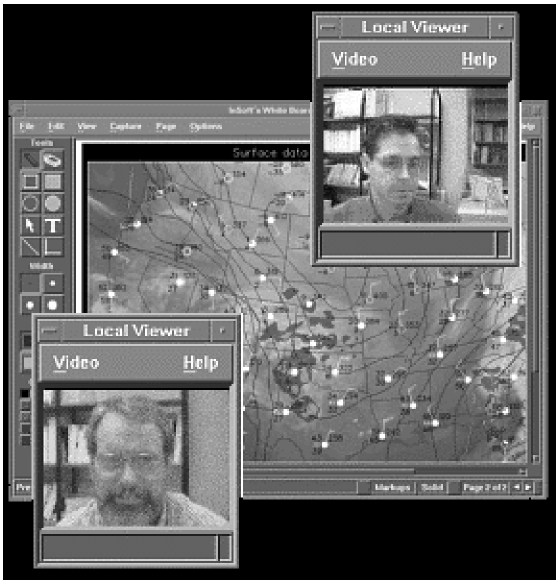

FIGURE 18.4 A METOC video conference.

Excepted from Bass et al., Software Architecture in Practice, First Edition (ISBN-13: 9780201199307)
Copyright © 1998 Pearson Education, Inc. Do not redistribute.

HYPERTEXT APPLICATION

The second type of interaction mode is person-to-hypertext over the WWW. The Web is used by the METOC Anchor Desk System to do the following:

- Store daily weather products such as specialized weather depictions, forecasts, and satellite images
- Help train people in standard operating procedures
- Help organize the weather products of the different service components associated with a crisis task force
- Download images from the Web that can be cut and pasted into computer presentation tools, such as PowerPoint
- Enter data so that observations from the field can be made available immediately by hypertext

METOC ANCHOR DESK SCENARIO

This section describes one of the METOC Anchor Desk scenarios that has been used in actual practice. This scenario will serve to make the concepts more concrete and will guide the discussion of the architecture and design of the METOC Anchor Desk System in later sections. We describe the situation and how the system might be used to respond.

A hurricane is approaching the Hawaiian Islands. In accordance with military procedures, there are sets of activities that occur when an extreme weather event is expected. During the normal course of events, there may be from 6 to 30 briefings of the commanding authority for each service. The weather officers of each service are all concerned with different aspects of the weather and can provide different insights. Normally, the senior weather officers of each service will collaborate before and after each briefing to their commanders, and these collaborations are done through the auspices of the METOC Anchor Desk.

During a collaboration, the following and other products available on the Web, might be called up for viewing at any time:

- East Pacific satellite image
- Close-up satellite image of the hurricane
- National Weather Service buoy data
- Latest U.S. Air Force reconnaissance plane reports
- Thirty-six-hour forecast
- PowerPoint slides of briefings by each service to their commanders

Not only are these products available to the METOC officers, but they are also available for retrieval directly by staff of the commander. In-place service personnel can, through video conferencing and a shared whiteboard, discuss their interpretations of the weather data and their probabilities. Computer forecast models can be run and their results shared immediately.

Excepted from Bass et al., Software Architecture in Practice, First Edition (ISBN-13: 9780201199307)
Copyright © 1998 Pearson Education, Inc. Do not redistribute.

18.3 Architectural Approach

The METOC Anchor Desk project made extensive use of three different emerging technologies. It uses the WWW (described as its own case study in Chapter 7), Common Object Request Broker Architecture, or CORBA (also described as a separate case study in Chapter 8), and collaboration systems.

The architecture of the METOC Anchor Desk System is not novel. In fact, that was one of its strengths; it created almost nothing from scratch. So, to truly understand the architectural solution, we must first understand the method that led to it: the Evolutionary Development Method. One of the principle tenets of this method is that architectures should be created through assembly of tailorable large-grained hardware and software components.

The code that was written for the system included Web authoring code (and scripts to generate it), high-level scripts, adaptation code (to modify components to suit its purposes), and conventional software (to provide weather objects to other servers). The system software components that were assembled were designed to work together on a variety of different computers.

The key architectural drivers behind the METOC project and the solutions adopted are shown in Table 18.1.

EVOLUTIONARY DEVELOPMENT METHOD

Evolutionary Development is applicable to development environments in which there is an authoritative and clearly defined user community where direct interaction with developers is possible. For reasons we shall see, it would not be effective in environments where there are users shielded from developers by layers of bureaucracy or in the development of shrink-wrapped systems where the user community is large and amorphous. The five paragraphs that follow describe the basic principles of Evolutionary Development.

Requirements and System Evolution Through Scenarios. Effective communication requires shared experience and shared vocabulary. Once an ini-

TABLE 18.1 How the METOC Anchor Desk System Achieves Its Quality Goals

Goal	How Achieved
Short time to initial demonstrable capability	Heavy use of off-the-shelf components
Developer productivity	Only writing tailoring/glue code
Interchangeable parts and interoperability	Loose coupling through use of CORBA and WWW/HTML
Portability	Use of portable commercial products and emulators

Excepted from Bass et al., Software Architecture in Practice, First Edition (ISBN-13: 9780201199307)
Copyright © 1998 Pearson Education, Inc. Do not redistribute.

tial shared vocabulary has been developed, developers and users can begin to communicate about the problem to be solved and the technological options available. First, the user describes a scenario in terms of the current solution and begins to appreciate the possibilities of the new technology. Much later, and after some hands-on experience with story boards and prototypes, the user is able to articulate that scenario in terms of a new technology. Then the discussions broaden so that when the technology is described, the user may reflect on new ways that the technology can be employed in the context of any number of user situations. The system evolves along with the scenarios. As the system is incrementally developed, the scenarios become more detailed. Mistakes in the scenarios and in the system are found at the earliest possible time.

Continuous End-User Involvement with Continuous Recalibration. One of the biggest problems in the development of large systems is that the system requirements are defined once at the beginning, and without consulting the users again or revisiting the requirements, the developers take a long time to build the system. Two things inevitably happen. First the user's needs change over time. Second, the user's perceived needs as originally expressed turn out not to be the real needs in the context of new technology when the system is delivered. The job of systems builders is to accommodate those factors in a meaningful way. Unfortunately, user involvement is often defined in terms of more frequent and more intense design reviews with customer representatives who are not actual users.

Evolutionary Development facilitates effective user involvement, including hands-on experience with a real product or prototype. By getting frequent user feedback on products or prototypes, change is accommodated by accepting the need for rework and minimizing its effect. When an errant path is pursued, or when a user need changes, it is immediately obvious to the developer and the cost of the change is minimized. The advantage of the user-centered approach is that there is no requirement to specify every detail before moving on to the next stage of implementation. Instead, we use feedback from one iteration to feed the directions for the next iteration. This continuous recalibration minimizes the effect of the many mistakes that are *inevitable* in trying to specify the requirements of a system.

Architectures Based on Assembly and Integration. Object technology and client-server architectures are on the verge of making prototyping and reuse work for real systems by providing an architectural infrastructure into which separately developed components can be plugged. The individual pieces of technology need to know less and less about one another to work together.

Sensible and Manageable Documentation. The cost driver of large systems is often its documentation. Various standards required by government organizations require voluminous documentation on all phases of a multiphase development. (A possibly apocryphal story holds that the newest U.S. Air Force cargo plane is incapable of carrying all of the documentation produced during its development.)

Excepted from Bass et al., Software Architecture in Practice, First Edition (ISBN-13: 9780201199307)
Copyright © 1998 Pearson Education, Inc. Do not redistribute.

Not only is this documentation expensive to produce in the first place, but it is even more expensive to maintain: A small user enhancement or change in operating environment can cause a ripple of changes through requirements, design, testing, maintenance, and training documentation.

The evolutionary approach reduces the initial documentation requirements and obviates much of the need to change the documentation because the inevitable blind alleys are followed to less depth. No detailed requirements document is written because the evolving system best defines the detailed requirements. In its place is a document that covers the concept of operations and scenarios of usage to set the scope of the work. Architecture, design, and training documents are still required. Test documentation is simplified because it is done incrementally with each delivery rather than trying to plan the testing for the entire system before the system is known. The user scenarios, when complete, can be the basis for much of the requirements, testing, maintenance, and training documentation.

Rework. Evolutionary Development is a powerful method to solve the problems it attacks but it is not a panacea. Rework is an important element of Evolutionary Development. Every iteration results in refined requirements and corrections to components that embody incorrect assumptions. This holds for the documentation as well as for the system itself. The key to making the process effective is to only go into depth on portions that are agreed on. This not only results in avoiding time going down blind alleys but also reduces the rework involved in tuning the portions of the system that are going to be completed in depth. Assumptions about the underlying infrastructure are revisited during every iteration. For portions of the infrastructure that are themselves going quickly through the ABC, rework needs to be done to accommodate new releases. We will see in a subsequent section how new releases of the WWW software required accommodations in the METOC Anchor Desk System.

18.4 Architectural Solution

This section describes the METOC architecture. Like all architectures, it was a response to specific quality requirements, which we begin by describing.

KEY ARCHITECTURAL DRIVERS

The architectural drivers of the system are derived from the goals of both the customer and the user. We identify the following in order of importance to the developers. Some of the drivers are hard requirements, as indicated by (H) in the list, that were levied by the customer for business reasons.

- Short time to initial demonstrable capability (H)
- Evolvability

Excepted from Bass et al., Software Architecture in Practice,
First Edition (ISBN-13: 9780201199307)
Copyright © 1998 Pearson Education, Inc. Do not redistribute.

- User productivity (H)
- Developer productivity
- Platform heterogeneity
- Interchangeable parts (hardware and software)
- Geographical distribution (H)
- Interoperability, with legacy tools, decision aids, and other anchor desks (H)

Short time to initial demonstrable capability is fundamental to the Evolutionary Development concept. Prototypes must be put in the hands of users to help define the requirements and to establish a working relationship between the user and the developer. We link short time to initial demonstrable capability with initial operational capability and evolvability. Initial operational capability is limited in its functionality by design.

A second driver is productivity on the part of the developer. A small team was attempting to provide a great deal of capability to the users. In any user-centered development, it is necessary to keep the user engaged by providing a stream of capability increments based on feedback from demonstrations. There was not time for long analysis–design–implement–test–demonstrate cycles. Unless they could show that they could produce something useful quickly and unless they could continue to produce increments quickly, the developers would have lost the attention and commitment of the user.

Platform heterogeneity was a driving factor because of legacy systems. There are three kinds of machines in the user environment: UNIX workstations, Macintoshes, and PCs. The architecture had to accommodate this mix of computing equipment either by making software run on all three or by emulating one machine on another. It would have been unrealistic to assume that the user community would adopt a single platform.

The need for interchangeable parts is closely coupled to the evolvability driver. Tight coupling of components makes a system rigid and inflexible. Loose coupling makes a system changeable and flexible. Adding functionality to system components or improving performance of system components without changing the architecture was a goal. Less important was maintainability. Because the system was built primarily from off-the-shelf components, there was little that was maintainable by the development team. Rather, the component parts would be maintained and upgraded by the original developers of those parts.

Geographical distribution of the users drove the architecture as well. The Pacific theater of operations covers a large portion of the world and users may be in every section of that theater and in other theaters around the globe.

Finally, interoperability was a driving factor. The METOC Anchor Desk System must interoperate with legacy tools, decision aids, and other anchor desks. With regard to legacy tools, weather reporting and forecasting capabilities have been developed over many years, mostly on mainframe computers. The METOC Anchor Desk System uses these legacy tools. The decision aids and other anchor desks, on the other hand, are relatively new.

Excepted from Bass et al., Software Architecture in Practice, First Edition (ISBN-13: 9780201199307)
Copyright © 1998 Pearson Education, Inc. Do not redistribute.

It is also instructive to list some of the notable nondrivers of the architecture. In particular, there was little consideration given to most of the typical runtime drivers of performance, reliability, safety, security, correctness, availability, or resource constraints. The reason that these were not architectural drivers was not because they were considered unimportant but because they were not under direct control of the developer. These runtime factors are largely dependent on the platform dependent components used to construct the system. For example, the safety, security, and correctness of the video-conferencing software is critical to the safety, security, and correctness of the overall system.

Certainly, the platform dependent components must be selected so as to make the system viable, but they are not drivers of the architecture. The interchangeable parts driver ensures that the system is not burdened with unacceptable components. Many times during the development, changes were made from one hardware component to another or from one software component to another to improve the system relative to one or more of these nondriving influences.

ARCHITECTURAL COMPONENTS

The small size of the development group and the need for delivering early initial capability dictated assembling the system from already-available components. In some cases this meant buying commercial software and in other cases using public domain software or shareware. The major categories of components were the following:

- Computers—UNIX workstations and PCs, including laptops
- Networks
- Video-conferencing software
- Emulators
- Web browsers
- Collaboration software
- Utility software

Computing resources included workstations and personal computers. Both categories are in widespread use in the METOC community and are accommodated in the system architecture. The workstations are all UNIX compatible machines and are primarily Sun Sparcs and Hewlett Packard TAC-3s. The PCs are both Macintoshes and IBM compatibles. Specialized hardware boards may be required on the computers to capture video images from a camera and display them on the screen.

The *network structure* is a four-tiered structure that is illustrated in Figure 18.5. At the weather center there is a 10-Mbit/second local area network (LAN). This is connected by a fiber-optic bridge to another 10-Mbit/second Operations Support System LAN for local use in Pearl Harbor. These LANs are connected to

Excepted from Bass et al., Software Architecture in Practice, First Edition (ISBN-13: 9780201199307)
Copyright © 1998 Pearson Education, Inc. Do not redistribute.

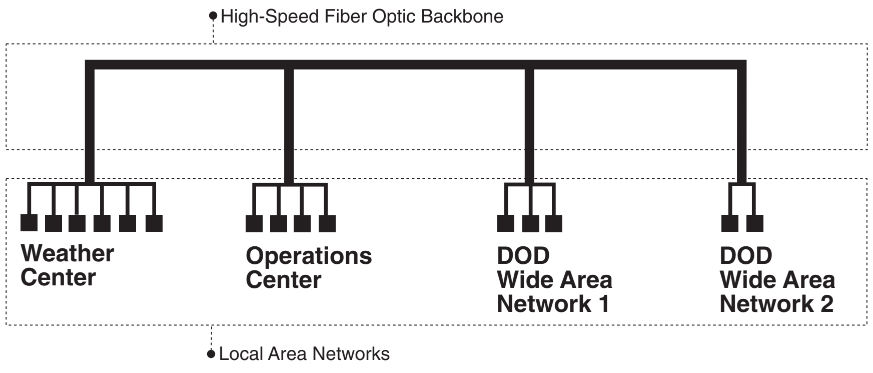

FIGURE 18.5 METOC network architecture.

preexisting Department of Defense wide area networks by T1 (1.54-Mbit/second) satellite links.The network architecture must not only provide the necessary connectivity, but there must also be enough bandwidth to carry out the communication and collaboration functions.

The *video-conferencing software* permits geographically distributed operators to carry on an audio/video conference and to collaborate over graphic and textual data.

Emulators are programs that run on UNIX machines to simulate Macintosh and PCs. Emulators, among other things, allow a UNIX computer to read and write Mac or PC disks.

Web-browser software permits users to easily access linked information on networks of computers. The browser presents information that contains links to other pages on different systems that can be invoked with a mouse click. These browsers support a wide variety of graphics formats and also permit the invocation of local computing resources, such as viewers and audio playback utilities.

Collaboration software is special-purpose software that allows two or more users to collaborate over an application.

Utility software is a catchall that includes a variety of functions—graphics scanning, graphics conversion, audio and video recording, fax software, and so on.

DESIGN OF THE METOC ANCHOR DESK SYSTEM

The overall structure of the METOC Anchor Desk organization is shown in Figure 18.6. This figure illustrates three major components, the major participants and suppliers (identified in the rectangular boxes), and the planning cells (shown in the circles). It illustrates the distributed, interconnected nature of the architecture, which is motivated by the need for the system to be geographically distributed.

Excepted from Bass et al., Software Architecture in Practice,
First Edition (ISBN-13: 9780201199307)
Copyright © 1998 Pearson Education, Inc. Do not redistribute.

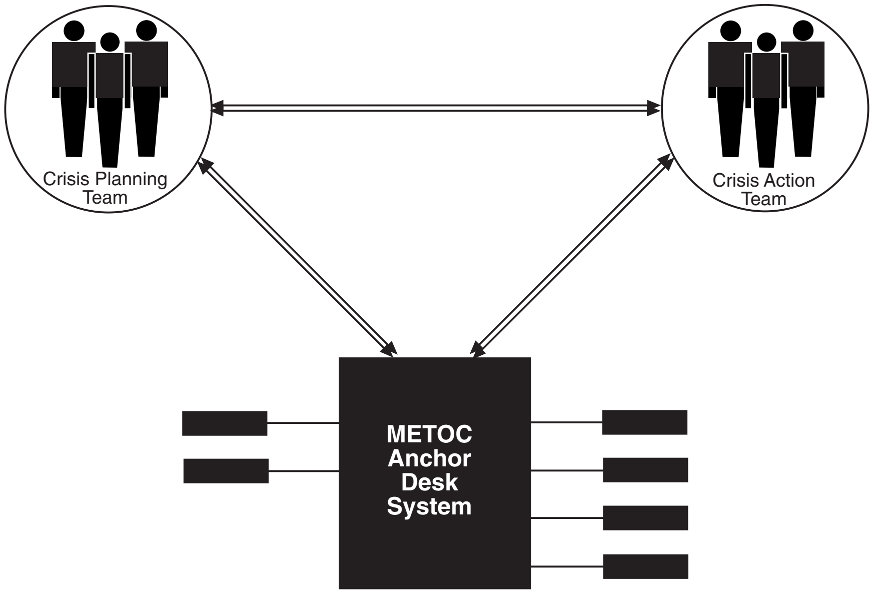

FIGURE 18.6 METOC Anchor Desk organization.

The key design decision of the METOC Anchor Desk System is not to disturb the legacy organization and the legacy systems that are in place and work well. The idea is to enhance, not to replace. From this decision come the derived requirements that address the specific computers and tools, how they are tied together into a global network, how the information space is structured, and how applications are distributed.

The design is concerned with the integration of the architectural components described in the previous section. Most of the integration is achieved by loose coupling. The integration among video-conferencing software, Web browsers, emulators, collaboration software, and utility software is primarily through UNIX files.

For integration among other tools in the project, an object request broker (ORB) approach is taken. To be effective, the system had to interface both with newly developed tactical decision aids and with legacy METOC applications. ORB interfaces were developed by other project teams for capabilities such as map and data servers. It was important not to have to reinvent component parts to provide capabilities. This design characteristic served the following architectural drivers: short time to initial demonstrable capability, developer productivity, interchangeable parts, and interoperability.

As a consequence of this design decision, there was little code that had to be written. Scripts were written for utility operations, and adaptations were made so

Excepted from Bass et al., Software Architecture in Practice,
First Edition (ISBN-13: 9780201199307)
Copyright © 1998 Pearson Education, Inc. Do not redistribute.

that software could run on a variety of hardware. Decision aids were written to be compliant to ORB-based standards.

The idealized version of the METOC architecture is shown in Figure 18.7. Here we use the orchestra metaphor, suggesting a loose connection between the conductors (director and choreographer) providing direction and the orchestra (gathering, serving, and analysis objects) performing according to the direction. This can also be thought of as a client-server architecture but with more insulation and flexibility between the client and server. In Figure 18.7 the clients are illustrated on the top and the servers on the bottom. The ORB is the substrate or mechanism used for interfacing the client and server. The services listed include storage and retrieval of the raw environmental data, data gathering, data analysis, data visualization, and mapping. We distinguish between conductor- and orchestra-level roles in describing this architecture.

One conductor-level role is the director. The director presents end users with a palette of available decision aids (tools) and permits selection from the palette. It maintains a registry of tools that form the universe from which palettes can be chosen for each user. Finally, it allows new tools to register themselves at runtime. New tools can then be made available to end users without recompilation.

Another conductor-level role is the choreographer. This role insulates the director from knowledge of elements at the orchestra level. Each of the choreographers has knowledge of one subset of the orchestra-level objects that are required to service one particular tool. By providing this level of indirection, the runtime extensibility of the director is enhanced and the localization of knowledge (to the choreographers) reduces complexity.

At the orchestra level, the data server provides direct access to environmental data. It insulates data-gathering objects from knowledge of underlying database management systems. The data gatherer knows how to gather and filter specific kinds of environmental data for a particular tool or class of tools. The

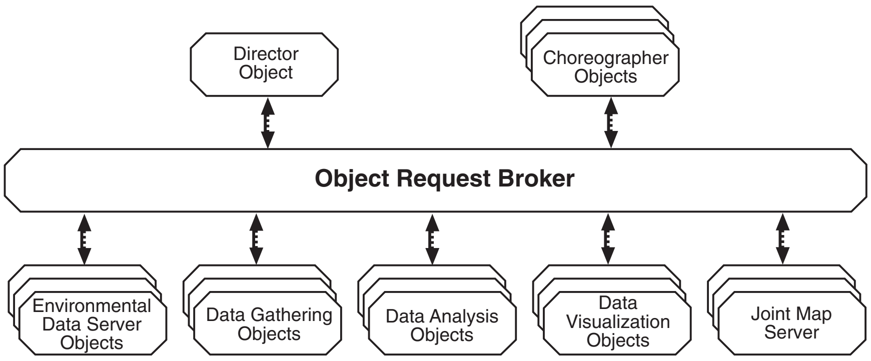

FIGURE 18.7 METOC Anchor Desk System architecture.

Excepted from Bass et al., Software Architecture in Practice, First Edition (ISBN-13: 9780201199307)
Copyright © 1998 Pearson Education, Inc. Do not redistribute.

data analysis service acts on gathered and filtered data to produce some analytical output for downstream processing. The visualizer renders the result of data gathering and analysis. Several visualizers may operate on the same data to produce alternative visualizations. The map server is utilized by some of the visualization objects to render the weather over a map. This map may show geopolitical boundaries, topographical features, or landmarks such as roads and airports.

In this architecture, a new tool may be introduced quite simply by adding a new choreographer object and, if required, new services such as an analysis or a visualization. This produces a great deal of flexibility and supports the short time to market and developer productivity drivers.

Another characteristic of the design was the choice of both commercial off-the-shelf products and public domain products that operated across many different platforms. In many cases more than one product would be necessary for a single function. The architectural driver in this case was the multiplatform requirement. For example, an early decision was made to use a commercial product, Communique, for video conferencing. This decision was made, in part, because Communique was supported on UNIX workstations and PCs. An alternative public domain product was only available on Sun systems. After the public domain product was adapted for HP machines, it became an alternate choice for implementing video conferencing for better integration with other project products. Netscape replaced Mosaic as the Web browser during the development due to its greater functionality.

RESULTING IMPLEMENTATION

The resulting implementation of the METOC Anchor Desk System is shown in Figure 18.8. In this version of the system, software runs on HP machines, a Sun, a desktop PC, and a portable PC. All the machines are running video-conferencing software, Web browsers, and frame grabbers. The HPs and Suns run collaboration software. Where required, these machines run Mac and PC emulators and decision aid tools.

The implementation started by using a commercial video-conferencing product to demonstrate the concepts and gather performance data but later switched to public domain software for compatibility with other projects in the program. The shareware did not work on HP platforms and had a limited shared whiteboard capability. The developers made source code changes to make the video-conferencing software run on the HPs and improved the whiteboard.

The Theater-Level Analysis, Replanning and Graphical Execution Toolbox (TARGET) is an integrated set of planning tools for supporting joint operations. It is designed to support distributed collaborative planning. One of the functions provided to TARGET by the METOC Collaboration System is the ability to place weather information in TARGET folders. To accomplish this, the developers wrote scripts in Tcl/Tk to automatically grab a screen window as a GIF image and then to ship it off to TARGET.

Excepted from Bass et al., Software Architecture in Practice,
First Edition (ISBN-13: 9780201199307)
Copyright © 1998 Pearson Education, Inc. Do not redistribute.

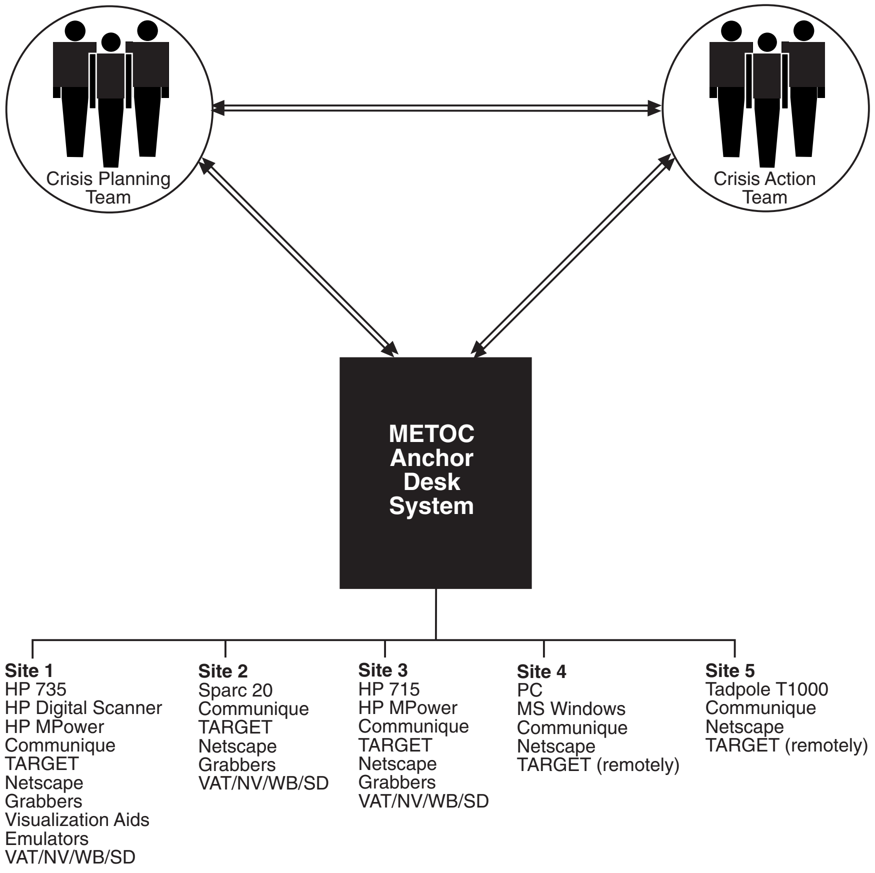

FIGURE 18.8 A METOC implementation.

The METOC Anchor Desk System permits collaboration using legacy tools as illustrated in Figure 18.9. From an appropriately equipped terminal it is possible to view and operate on a variety of objects that previously required three computers. In the METOC system, these operations can take place in separate windows on the same machine by using public domain and commercially available emulators.

In addition, many legacy systems use Macintoshes and PCs. Macs are used mostly for presentations, while PCs are used for various decision aids, general data processing, and file transfer. MacCIDSS and PowerPoint are examples of two of the Mac applications, and Optimal Track Ship Routing and Enroute Weather Forecast are two of the PC applications. With Mac and PC emulators on the same machine as the video-conferencing tools, it becomes easy to do analysis

Excepted from Bass et al., Software Architecture in Practice, First Edition (ISBN-13: 9780201199307)
Copyright © 1998 Pearson Education, Inc. Do not redistribute.

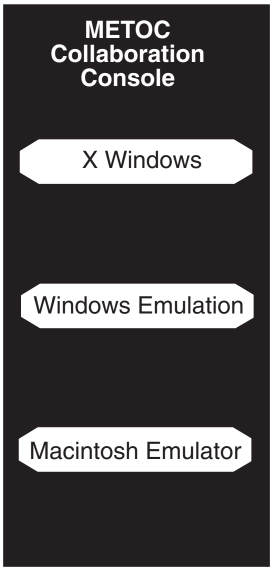

FIGURE 18.9 Tools for collaboration.

and then paste the results onto a whiteboard and share it with the conference. Conversely, when someone else shares a result, it is possible to grab it from the whiteboard and paste it into a PowerPoint presentation.

Figure 18.10 illustrates the kind of data that is available from the World Wide Web and the parties that access and supply that data. Two important decisions about using the Web are what information is to be stored and how it is to be presented. The structure of the information in the Web remains relatively constant from day to day, but the data is perishable and changes by the minute.

Considerable effort was spent designing and writing dozens of Web pages using HyperText Markup Language (HTML). Tcl/Tk scripts automate the moving of data from an "incoming" area on disk to the Web pages so that creating a new set of data in the Web is a simple one-line operation for a METOC user. This ease of use is central to getting the user to welcome the adoption of the new technology.

One of the collateral uses of the Web is as a repository for METOC procedures. The METOC operations floor is a beehive of activity 24 hours a day. There are satellite images arriving, forecasts being written, advisories being sent out by fax, etc. Hundreds of operating procedures appear in notebooks taking several feet of shelf space. Many of these procedures can better be represented in Web pages. Each step in a procedure may have links to more detailed instructions, illustrations, audio and video clips, and even to tools to be used when performing the step.

Excepted from Bass et al., Software Architecture in Practice,
First Edition (ISBN-13: 9780201199307)
Copyright © 1998 Pearson Education, Inc. Do not redistribute.

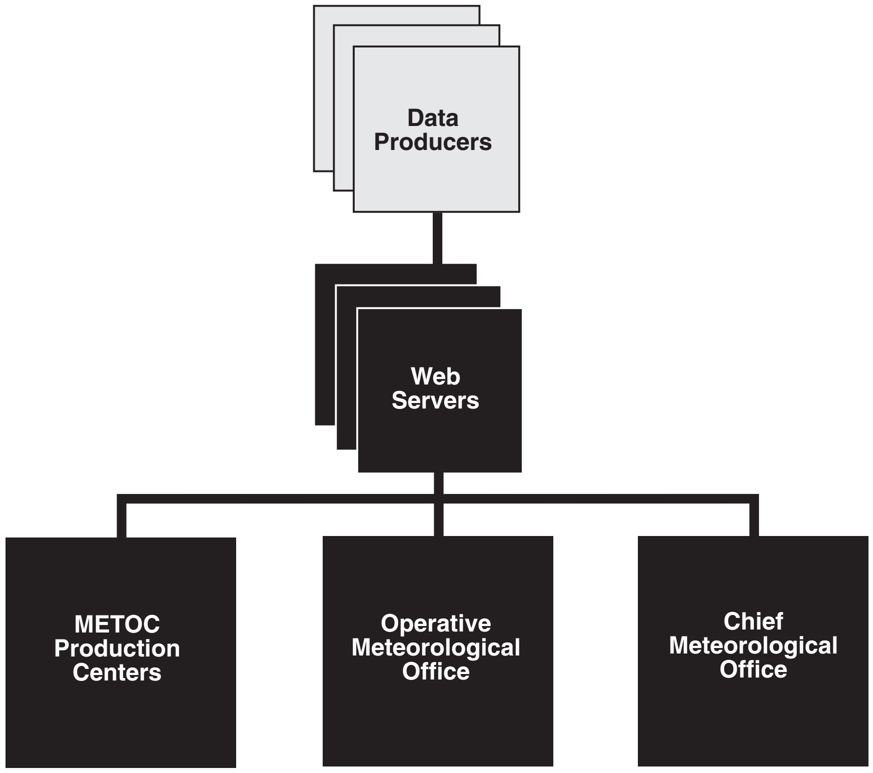

FIGURE 18.10 METOC products web.

As an example of a collaborative decision-aiding application that can run using the METOC Anchor Desk System, the developers implemented a weather specialist that plugs into a legacy map server. This weather specialist overlays pressure, wind, precipitation, cloud cover, and temperature on a given geographical area displayed by the map server.

The map server has several subservers (specialists) that it communicates with to draw maps. The map server uses a C++ Specialist class that was specialized to handle weather displays. CORBA allows the weather specialist object to register the weather specialist with the ORB. The specialist then waits for client requests. When the map server gets a request to show weather, it asks the ORB for a weather specialist object. Then it asks for the needed information from the object and displays it.

Thus, weather functionality was added to a weather-ignorant map server; the weather software needed no map-specific details other than coordinates. Furthermore, the map server did not have to know anything about weather and only provided

Excepted from Bass et al., Software Architecture in Practice,
First Edition (ISBN-13: 9780201199307)
Copyright © 1998 Pearson Education, Inc. Do not redistribute.

TABLE 18.2 Lines of Source Code Written for the METOC Anchor Desk System

Weather Specialist	C++	3000
GrabWindow script	Tcl	25
GrabWindow script	C	350
SendImageToTarget script	Expect	92
SendImageToTarget script	C	50
METOC Product Web	HTML	1215
METOC Product Web	Tcl	403
METOC Procedures Web	HTML	10014
METOC Developers Web	HTML	5154
Total lines of code		20303

an interface written in CORBA's Interface Description Language. This loose coupling of the implementation components through the use of object technology facilitated the implementation in terms of the productivity and interoperability drivers.

Table 18.2 gives the approximate number of lines of source code that were written by the developers, as opposed to represented in an off-the-shelf product, and the language used. Note the large number of languages and the small number of lines of code. There are five languages represented—hypertext, scripting, and general-purpose languages—and each is chosen for a specific task. The figures for HTML are not good indicators of effort since many of the lines in those instances are data giving textual information and are copied from other sources. Not represented in this table is any effort involved with creating graphic images.

Another item of note is the knowledge and background required to produce the code. C, C++, and Tcl were written by software engineers. HTML was written by graphic designers who were either undergraduates or recent graduates. Thus, the bulk of the code was written using preexisting text or with relatively inexperienced personnel. On the other hand, the creation of graphical images is not enumerated in the table, and this requires a type of skill not usually found in software engineers.

18.5 Summary

In Chapter 8 we saw object request brokers from the point of view of the developers of the ORBs. In this chapter we have described a system constructed on top of an ORB from the perspective of a user of the ORB technology. This case study provides evidence that many of the claims of Chapter 15 and Chapter 17 about the ability to integrate legacy systems and incorporate systems on diverse platforms are achievable in practice.

Excepted from Bass et al., Software Architecture in Practice,
First Edition (ISBN-13: 9780201199307)
Copyright © 1998 Pearson Education, Inc. Do not redistribute.

In Chapter 13 we presented architecture as the foundation for component-based system development. The architecture forms a skeleton into which separate but cooperating components are plugged. This case study provides evidence that component-based development using off-the-shelf pieces can succeed, be economical, and still satisfy the system's requirements.

Finally, Chapter 7 presented the World Wide Web and a unique set of motivating requirements that led to its design. This case study illustrates how those features of the Web were put to use as the backbone of a widely distributed decision-making system. When the project began in 1993, Web technology was still relatively immature. However, it was the best available to meet the project goals, so the developers earnestly adopted it and began the process of making that technology suit an operational military context. This involved designing and deploying mission-critical Web pages of environmental information. This work continues today because the Web pages are constantly evolving to meet new needs and deliver new functionality.

Since 1992, Web technology has changed significantly. Some of these changes affected the appearance of Web pages that the METOC developers designed. For example, the advent of tables in Netscape significantly improved the ability to present tabular information. Prior to tables, a complex scheme of transparent in-line GIF images was used to attain modest control over horizontal and vertical white space. Tables provide a much more robust and flexible palette from which to design and implement. Initially, the developers adapted earlier pages to use simple tables. As understanding of tables grew, they dealt with new issues of page layout and implementation that had not existed previously.

Adaptation, experimentation, demonstration, and evolution are recurring themes in this project, and all are made possible by the robust architecture and the loose coupling among the components that it provides.

18.6 Discussion Questions

1. How dependent is the Evolutionary Development method on the existence of technologies such as CORBA and the WWW? Could it exist in the absence of such technologies? Or, more generally, what are the ramifications of the technical environment on the ways in which the ABC is realized?

2. In Chapter 13, we discussed building domain-specific languages as an approach to architecture-based development. How does this approach and the Evolutionary Development method relate to each other?

3. What architectural styles do you recognize in the METOC architecture?

Excepted from Bass et al., Software Architecture in Practice, First Edition (ISBN-13: 9780201199307)
Copyright © 1998 Pearson Education, Inc. Do not redistribute.