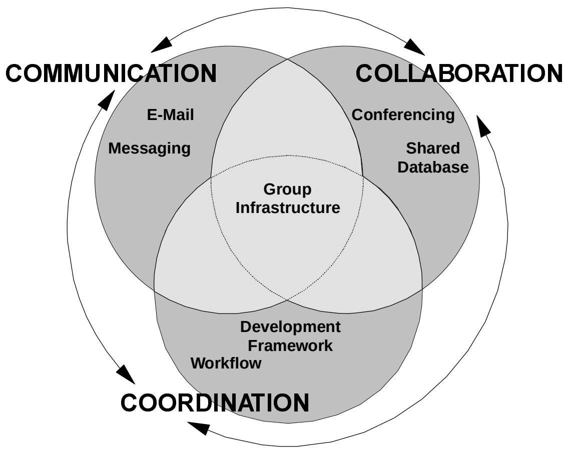
Groupware - Communication, Collaboration, Coordination

Executive Summary



[<img src= "circle4.gif">]

In many ways, groupware defies definition. Nonetheless, this software category has captured the attention and imagination of Information Technology (IT) professionals, line of business managers and end users, not to mention software suppliers. While most businesses have not developed a clear definition of groupware, they are keenly aware that leveraging the knowledge of employees and trading partners is the key to survival and success. Furthermore, businesses know that a clear competitive advantage lies with those who can effectively manage and exploit their intellectual assets.

Most definitions of groupware tend to focus on singular technologies with relatively narrow design centers. Not surprisingly, suppliers of products centered around communication -- "pushing" information out into an organization -- view messaging as the core technology for groupware. Likewise, suppliers of products centered around collaboration -- sharing information and building "shared understanding" -- tend to view computer conferencing and shared databases as central to groupware. Those with products aimed at assisting individuals and groups in the coordination of complex tasks involving a rich mix of delegation, sequential sign-offs, etc., are apt to view application development tools that support task and workflow automation as the *sine qua non* of groupware. It's because groupware is at the convergence of what were previously considered independent technologies (messaging, conferencing, workflow, etc.) that there is so much confusion about its definition and scope.

As obvious as it may seem, if we start from a belief that groupware should help individuals work together in a qualitatively better way, we find that groupware represents an integration of these technologies. This book uses a simple framework for group work, based on three categories:

- **Communication** - *rich electronic messaging;*
- **Collaboration** - *facilitating a rich, shared, virtual workspace;* and
- **Coordination** - *adding the structure of business processes to communication and collaboration, so as to implement an enterprise's policies.*

Through closer examination we will determine the conditions under which each technology model breaks down when used by itself. From this we learn that group applications require rich combinations of technologies. Furthermore, what makes a groupware platform powerful is its ability to support the dynamic movement *between and through* these three modes of group work: communication, collaboration and coordination.

Thus, groupware is not a laundry list of features and functionality, but is instead a platform that simply and elegantly mirrors this convergence. A groupware platform, therefore, is represented by the integration of three primary technologies:

- An **object store** in which corporate knowledge -- messages, documents, forms, memos, reports -- can be housed and managed.
- A **distribution and access model** that allows users to easily locate and disseminate information.
- An **application development framework** that leverages the native underlying services of the object store and distribution/access model.

Of course, a groupware infrastructure must take into account the general requirements of workgroup environments. Specifically, these include:

- **Integration with external resources.** The point of origin for workgroup information is often external to the groupware environment (i.e., desktop productivity tools, relational databases, etc.).
- **Platform independence.** While groupware applications often begin as departmental implementations, many eventually result in company-wide deployment. Platform independence is critical to ensuring universal use and investment protection.
- **Mobility.** A groupware infrastructure must be capable of supporting many geographically dispersed sites, including home, laptop, and notebook computers.
- **Inter-enterprise applications.** As businesses begin to rely on customers and trading partners as essential players in the automation of business processes, the ability to seamlessly extend the application -- from the start or added as an afterthought -- is an important part of a groupware infrastructure.

No business process application can be written that fully anticipates every situation. No matter how many exceptions and special cases are accounted for, people will discover new needs as they explore an application's depths and as new business situations present themselves. Thus, we conclude that any system designed to create, manage and leverage corporate knowledge is, by definition, of enterprise scale, and therefore must meet these criteria:

- It must support the full breadth of client, network and server operating systems.
- It must support mobile and remote workers.
- It must support seamless inter-enterprise interactivity

A groupware system that is architecturally correct in the sense that it supports the convergence of communication, collaboration and coordination is nevertheless doomed to failure on an *enterprise scale* if it does not also deal with the pragmatic realities of nomadic workers and inter-enterprise communication.

Updated: 07.11.95 22:10:00

Groupware - Communication, Collaboration, Coordination

Introduction

Knowledge is the only enduring asset of an institution. The ability to capture and manage that knowledge is critical to the survival and success of any organization, large or small. A category of software called groupware has emerged as an important technology that enables companies to create, share and leverage corporate knowledge. The implementation of a groupware infrastructure is the challenge that Information Technology (IT) executives face as users demand a broad compendium of groupware services. In fact, successful groupware implementations depend upon both the availability of the enabling technologies and the commitment of IT management to implement the necessary organizational and system infrastructure. Successful IT organizations will develop this infrastructure on a schedule that matches the oncoming demand of end users.

The term "groupware" is much used, little understood, and frequently the source of confusion and skepticism. This regrettable state of affairs is the symptom of two underlying tendencies. First, suppliers, users and observers have naturally tended to consider groupware as the sum of its applications, with little thought to appropriate or optimal technology and infrastructure. Therefore, it has been difficult to derive a consistent understanding of groupware technology from applications as disparate as electronic mail (e-mail), group calendaring/scheduling, forms routing, workflow automation, computer conferencing, bulletin boards and video conferencing systems, among others. In many cases, applications not originally designed as groupware have been retrofitted or stretched beyond their design center to meet a business need.

Secondly, discussions of groupware tend to focus around singular technologies with relatively narrow design centers. Suppliers have typically built products based on the technological roots most familiar to them. Not surprisingly, suppliers whose products center around *communication* -- pushing information out into an organization -- view messaging as the core technology for groupware. Likewise, suppliers whose products center around *collaboration* -- shared information and building "shared understanding" -- likely view computer conferencing and shared databases as central to groupware. Those with products aimed at assisting individuals and groups in the *coordination* of complex tasks involving a rich mix of delegation, sequential sign-offs, etc., view application development frameworks that support task and workflow automation as the center of groupware.

In truth, a complete groupware infrastructure not only supports these three modes of group work, but creates synergy among them, producing a whole greater than the sum of its parts.

This paper examines these essential dimensions of groupware -- communication, collaboration and coordination. It discusses the stand-alone technologies that have been used to support them -- messaging, shared databases, and workflow automation -- and examines the conditions under which each of these models breaks down when extended beyond its design center. The paper then describes how a complete groupware infrastructure represents the convergence of these otherwise distinct technologies.

Debates about infrastructure can sometimes become dogmatic, and often veer towards architectural purity at the risk of ignoring the practical realities of how individuals, groups and organizations actually use applications to capture, manage and leverage enterprise knowledge. In fact, the short history of groupware applications has shown that users themselves will stretch a technology to meet altogether different needs than those for which the application was originally intended. In this paper, we endeavor to balance the weight of architectural requirements with an appreciation for the practical issues of how groups really work. The reality that users themselves will extend technology in unanticipated ways is actually one of the most important elements of a groupware infrastructure: it must be flexible enough to allow users to bend and extend it to their specific needs, rather than rely on solutions as defined by others.

It will become apparent that there is a natural synergy between communication, collaboration and coordination in the evolution of groups and the technology that supports them. Specifically, messaging and shared database technologies are the fundamental basis for these three modes of group work and the integration among them. Accessible through application development tools and by end users alike, a shared database architecture provides workgroup members with the ability to design, manage and maintain group processes. Likewise, messaging provides a ubiquitous store-and-forward transport as the fundamental means for communication between people and applications. This dual architecture will be critical to the flexibility and customizability of the environment.

This paper also addresses the practical issues of group work: the need to support remote and mobile workgroup members, heterogeneous client, server and network operating environments, as well as the growing trends toward inter-enterprise implementations between customers and trading partners.

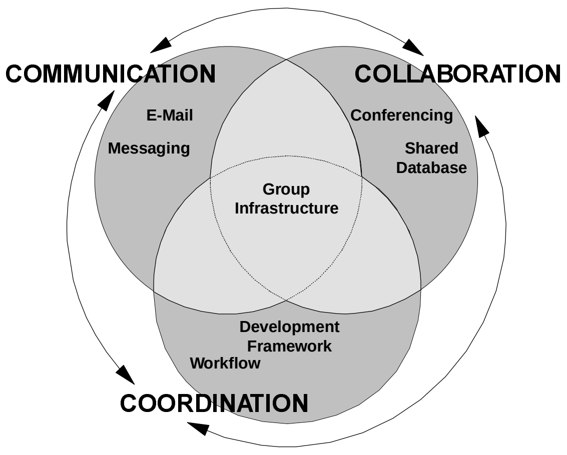
It is important to note that this paper does *not* purport to serve as the final word on the subject of groupware applications. It has been groupware's dynamic and evolving nature that has contributed to the lack of consensus on its definition in the first place, and there is no reason to expect the pace of change in the use of groupware to slow anytime soon.

However, while users will continue to apply groupware in unanticipated and innovative ways, the architectural underpinnings of the groupware infrastructure -- the focus of this paper -- are likely to remain constant into the foreseeable future. The paper concludes by offering architectural guidelines for building a groupware infrastructure.

Updated: 02.11.95 18:25:52

Groupware - Communication, Collaboration, Coordination

Defining Groupware



[<img src= "circle4.gif">]

Ask ten people -- CIOs, IT managers, end users, and software vendors -- for a definition of groupware, and you will no doubt receive ten different answers. The only common ground that most respondents will share is that they are not exactly sure where groupware begins and ends.

Definitions of groupware that go beyond "software that supports group work" usually focus on a single aspect of group work and a key application that supports that kind of work. Most users' experience with groupware has been piecemeal: they have used one or several groupware applications, but have no context in which to place them. Thus, users of e-mail and mail-enabled applications are naturally inclined to view groupware through the lens of electronic messaging. Users of forms routing products are naturally inclined to think of groupware as a function of workflow automation. And users of electronic conferencing systems or the World Wide Web are apt to consider shared access to information the root of groupware.

These different views arise because groupware, in fact, has its roots in three distinct, but increasingly overlapping, application areas: electronic messaging, information management, and workflow/process automation. Each of these technology domains has given rise to a number of popular groupware applications: e-mail, electronic conferencing and bulletin boards, and forms routing and tracking. The most prolific of these is electronic messaging; for this reason it is often viewed as the cornerstone of groupware.

However, if we step back from these myopic views of tasks performed within the workplace and take a broader view of how people really work, we find people moving from one of these work situations to another, changing modes of work, changing workgroup affiliation, juggling sets of only loosely related tasks, all of which require action during the workday, and few of which come to completion -- or even a recognizably stable state -- by the end of the day or the end of the week. We will find people picking up the phone to ask a quick clarifying question -- and often find that question left unanswered when the call can't be completed. We discover task forces forming and closing down -- new groups meeting in conference rooms to determine how they will work together and more established groups barely having to finish sentences yet communicating their ideas in phrases and gestures.

When we look at groups in this light, it is clear that groupware can't be defined as a single technology or a collection of applications. Groups' needs change over time, and every group is different from others. In order for a groupware system to be effective, it must be capable of supporting *all* modes of group work. Specifically:

- Electronic messaging is an effective tool for notification and a clear match to most communications needs. Messaging is such a powerful solution that users have begun to employ message stores as persistent data stores. Further, users have stretched its functionality by using mailing lists to support group meetings, or the co-creation and revision of common documents.

- However, bulletin boards and computer conferencing systems (shared databases) are used to provide a much more coherent, common view of group interaction, and these groupware applications have become commonplace in many organizations.

- Yet, systems where most information for groups is stored in databases start to suffer from the inherently passive nature of conventional database technologies. Each participant in a group has to take responsibility for finding information and scheduling their actions based on changes to that information. Thus, the need for a *coordinated* use of messaging for notification and database technology for shared information has emerged. The integration of messaging and shared databases in a single solution has evolved as a result.

- A workflow system based simply on forms which are routed from person to person again leaves each person in the group on their own, with no overview of the process. On the other hand, a workflow system based on a database storing the key materials and their status can provide full context for anyone who needs it, using messaging primarily for notification. Here, groupware represents the convergence of messaging, information management and workflow automation by meeting the evolving needs of groups to create, share and leverage corporate knowledge.

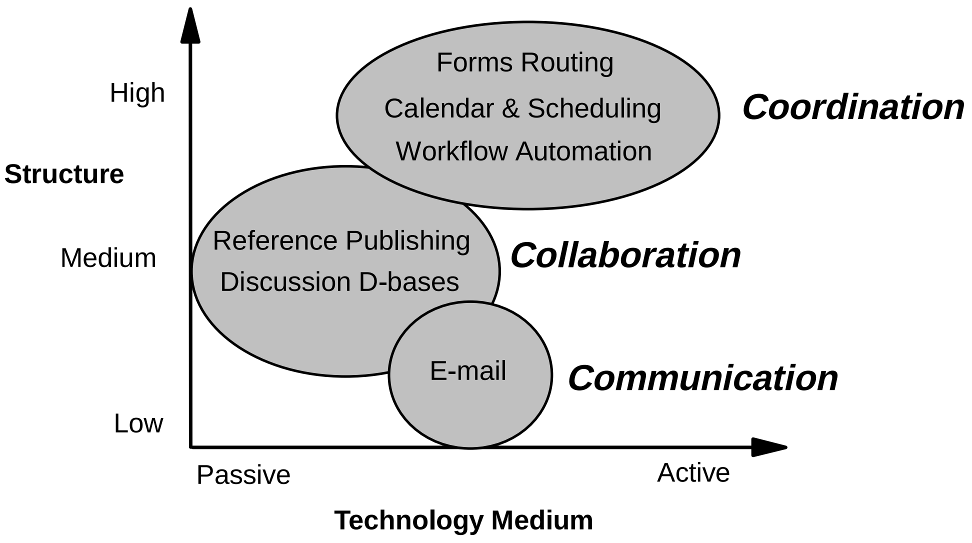
Looking at groupware from a more organic perspective, it's clear that information and knowledge are shared in support of three primary functions: *communication, collaboration, and coordination.* There are two dimensions that characterize the role that technology plays in facilitating group work: the degree of structure imposed by technology, and the passive/active role that technology plays in guiding the group work.

- The first dimension deals with the varying degree of structure required in group work. This may range from situations where information is distributed in an *ad hoc* fashion (i.e., sending an e-mail message to a group) to more highly structured processes where the steps are pre-defined and deterministic, such as routing a purchase order requisition.

- The second dimension addresses the passive/active relationship between the technology medium and the individual or workgroup. That is, passive applications leave control in the hands of the user or the workgroup, while active applications play a more proactive or directive role by controlling the flow of group work. For example, a shared database system that allows users to navigate a discussion thread is passive, whereas a system that actively monitors a process and notifies the user of an event is active.

The diagram below is useful in understanding how various groupware applications correlate to these dimensions, as well as their relationship to each other. Communication, collaboration and coordination systems each have their own unique characteristics. Understanding the design point of each area is important in identifying the criteria for technologies and applications in each space. *Taken together, these criteria form the basis for evaluating a comprehensive groupware architecture -- seamless support for communication, collaboration and coordination at any time, in any place.*

Our definition of groupware is simply this -- tools to enable people to work together through communication, collaboration and coordination. In fact, what makes a groupware platform powerful is its ability to support the dynamic movement between communication, collaboration and coordination. The following chapters examine these areas in terms of their group requirements and the technologies that support them.

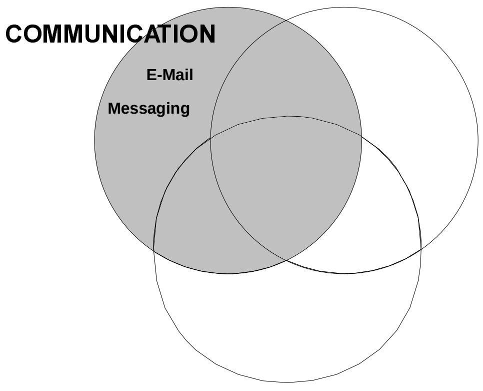


Categories of Groupware

Updated: 07.11.95 22:10:01

Groupware - Communication, Collaboration, Coordination

Communication



[<img src= "circle1.gif">]

Communication is the transmission of knowledge. In a business enterprise, colleagues communicate with each other in all sorts of ways: in formal meetings and presentations, through interoffice memos, over the telephone, and in informal hallway meetings. The information and knowledge imparted in these interactions takes the form of both verbal (written and spoken) and visual (images, drawings, body language) communication.

The role of a communication system is that of a passive electronic medium for transmitting information. Variables such as time, place and number of participants determine the most appropriate communication system in any given situation. Same time, same place, one-to-one interaction represents the simplest form of communication. When the combinations of time, place and number of participants increase, complexity is introduced. For better or worse, it is the most complex combination of dimensions through which most corporate knowledge is created, shared and leveraged.

Increasingly, individuals find themselves relying on electronic mail for interpersonal communication within and beyond the enterprise. While e-mail has improved the efficiency and accuracy of communication for some, it poses challenges to others, including users and network and systems administrators.

Updated: 07.11.95 22:09:59

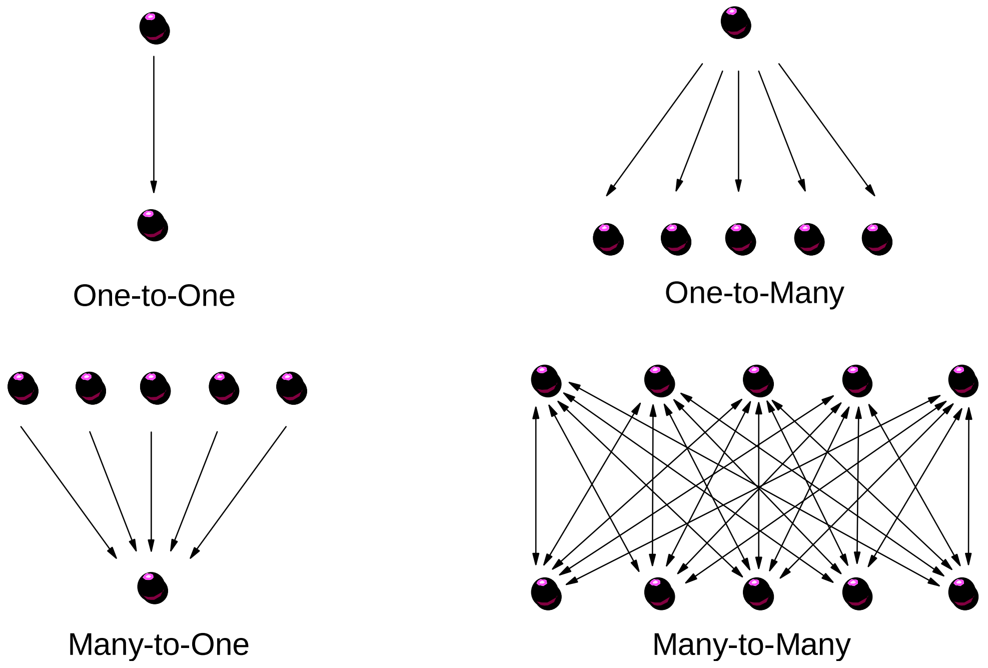
Groupware - Communication, Collaboration, Coordination

Electronic Messaging: A Technology for Communication

Electronic messaging is the store-and-forward transport of electronic objects among people, among people and applications, and among applications. The design point of electronic messaging is the asynchronous transmission of messages from one place to another. Messages can contain either simple or complex information, and they can be delivered to specific individuals or groups. Messaging supports different-time, different-place information sharing by virtue of its store-and-forward, or "push," model of transmitting or moving information. That is, information is "pushed" from the sender to the recipient.

Electronic messaging's store-and-forward transport system distinguishes it from other communication technologies. The store-and-forward transport is used to move, or "push" an object from one point to another along a number of intermediate points (i.e., from post office to post office) until delivered to the ultimate recipient. Messaging provides asynchronous connectivity because the sender and receiver need not be synchronized in time. Therein lies the real advantage of store-and-forward processing.

Messaging is credited with revolutionizing one-to-many communication. Naturally, this quickly leads to many-to-many communication. As depicted by the "web" of point-to-point paths in the diagram below, the use of e-mail for many-to-many communication has increased e-mail volume exponentially. As we shall see, this transition in communication is not nearly as simple a move for store-and-forward messaging.



Forms of Communication

Updated: 02.11.95 17:34:54

Groupware - Communication, Collaboration, Coordination

The Message Store

Traditionally, the message store has served as the temporary holding place for messages as they are being routed to their destination. Over time, however, as messaging became widely used for one-to-many and many-to-many communication, the message store has become the *de facto* "container" for large quantities of corporate information. That is, the temporary store designed for *ad hoc* messages has been used as a semi-permanent store not only for short-lived memos, but also for documents with an indefinite life cycle.

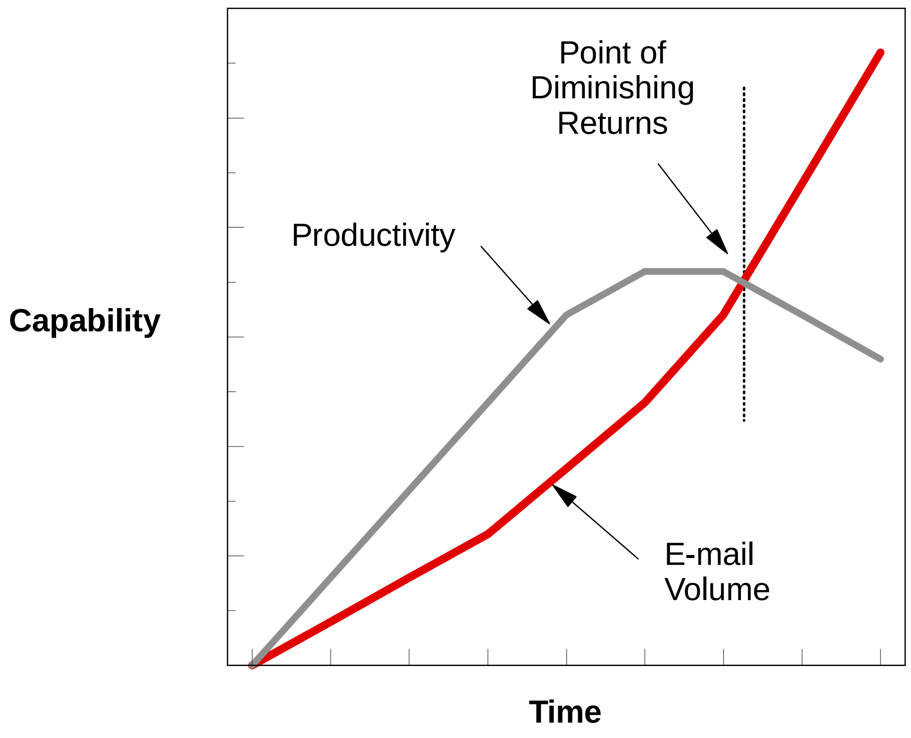
Because the original design center of a message store is as a *temporary* store, it has been optimized for delivery and retrieval of messages. This model assumes that the documents and objects that are routed using the e-mail system will be stored and managed elsewhere (e.g., the hard drive of a desktop computer, or a LAN-based file server) by the senders and recipients. *The message store was not designed for the persistent storage and management of information.*

Nonetheless, messaging systems have become the resting place for more and more corporate information. As a result, users and suppliers have explored ways to manage, manipulate and further automate the use of this information through various forms of foldering, rules-based processing and application development. Messaging APIs (Application Programming Interfaces) have emerged as the way to programmatically use the messaging system to move objects from a user to an application, from an application to a user, and among applications. Examples of this include mail-enabled desktop applications, group calendaring and scheduling, and forms routing applications. Messaging systems have become a virtual communications "hub," resulting in a significant information management problem that messaging systems simply were not designed to handle.

Despite the immense success of messaging as a communications vehicle, users are now beginning to complain that the technology is becoming increasingly unwieldy. There are two aspects of messaging that make it challenging for enterprise information management. First, users are unable to manage the ever-growing volume of messages. Second, the lack of sophisticated information management tools for manipulating the information contained in e-mail messages is becoming a larger and larger problem.

E-mail is the victim of its own success. Its ease of use and widespread adoption has led to an explosion of e-mail traffic. This results in users losing information, and accumulating large backlogs of unanswered messages. Valuable time is spent reading and sifting through messages of only marginal relevance. Some users have become so overwhelmed with the avalanche of daily e-mail messages that they have begun to consciously ignore many messages altogether! This unfortunate situation has a negative impact on both individual and corporate productivity, resulting in lost information and reduced customer responsiveness, and ultimately affecting the bottom line.

Furthermore, the problem will only get worse. The increasing capability of e-mail systems (e.g., the ability to send a 50 megabyte file to 10 or more users) will further exacerbate the problem. The graph below displays the relationship between message volume and productivity. Initially, there is a positive impact on productivity because people are getting the information they need faster. However, as the volume and complexity of information increases, users reach the point of diminishing returns in productivity. Eventually, this increased volume of information can have a negative impact on individual productivity, primarily because the information we need is co-mingled with the information others think we need or want us to have. In response to this, features such as rules, filters and hierarchical folders have been implemented to assist us with e-mail information management. At this point, any increase in productivity gained from getting information faster is lost in trying to sort through and access the information that is relevant.



Productivity and Information Overload

This phenomenon of "information overload" and its impact on productivity was recognized some time ago by the operations research community. In order to better understand this, it is important to distinguish between the benefits derived by both senders and recipients of e-mail. As senders, e-mail accelerates our ability to distribute information. As recipients, we benefit by receiving information more quickly. If the rate of information delivery exceeds our ability to absorb and manage it, we reach the point of diminishing returns. In other words, e-mail effectively solves the distribution problem for the sender, but ultimately creates an information management quagmire for its recipient.

Consider this example. A manager sends an e-mail message to a dozen people requesting feedback on a project. If all the recipients send all of their replies to all of the original recipients, the number of messages easily multiplies beyond the ability of a single recipient to track and manage the message thread. Moreover, there is no mechanism to help the participants identify which replies are in response to which messages. The greater the degree of interaction and the greater the number of participants, the more difficult it is to manage the dynamics of who said what in response to whom. This represents information overload and loss of the context of the information itself.

Getting to the root of this problem requires a closer look at the requirements of many-to-many communication and the need for an information management model. Many of the problems associated with overburdened messaging systems can be traced to their essential characteristic: support for unstructured communications through a "push" model. That is, the model, while effective in supporting unstructured communication, fails to effectively support more complex levels of interaction. In contrast, technologies designed to support the complex interactions of many-to-many collaboration use an entirely different model of information distribution and management.

Updated: 31.10.95 21:44:20

Groupware - Communication, Collaboration, Coordination

Conclusions

So far, we have isolated the area of communication in order to better understand its relation to other aspects of groupware technology. We've determined that e-mail is an effective medium for one-to-one and one-to-many forms of communication. Due to the lack of structure and volume of information being pushed, it appears that many-to-many communication quickly becomes unmanageable in this environment.

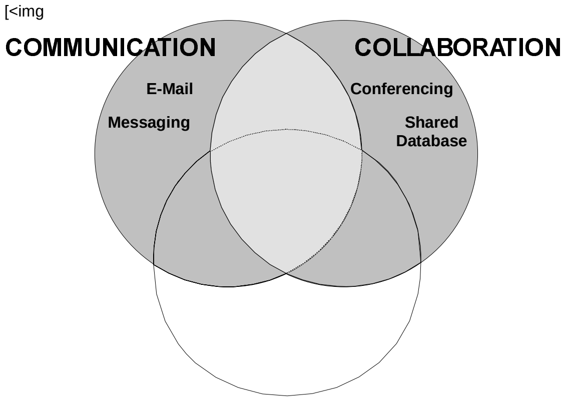
In determining the appropriate technology for resolving these issues, it is important to distinguish between information delivery and information management. As a store-and-forward transport, messaging is effective for information delivery. In order to resolve the information management problem, we must look to other technologies.

Updated: 31.10.95 21:45:18

Groupware - Communication, Collaboration, Coordination

Collaboration

[<img

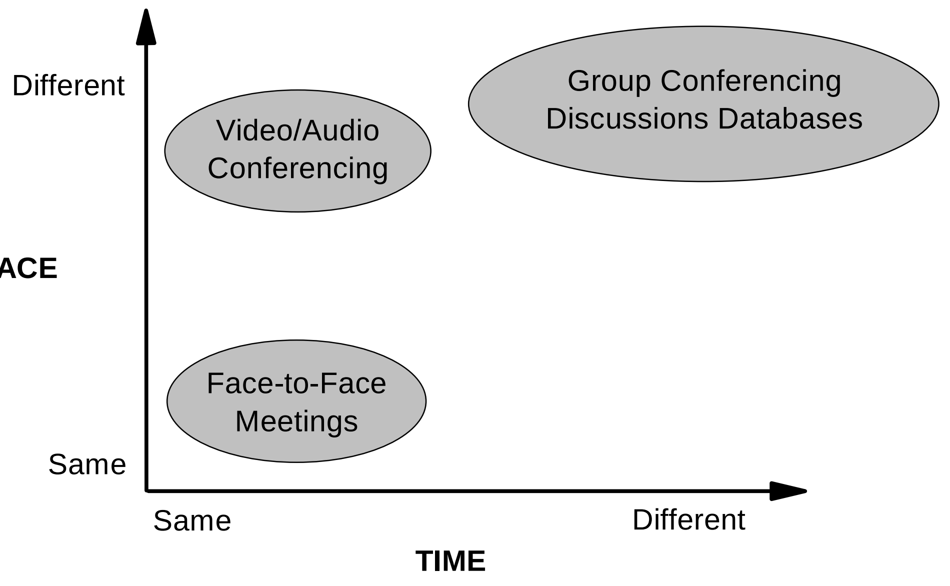


src= "circle2.gif">]

Collaboration relies on a shared space. It may be a room, a blackboard, a napkin, or a shared on-line space. Shared space serves as a touchstone for the act of collaboration, and it is essential as a medium to manage the ambiguity inherent in human interaction. In effect, these shared spaces are the collaborative tools that provide a context in which the whole of the relationship is greater than the sum of the individual participants' expertise.

Collaboration can be between two people, or can take the form of many-to-many information sharing. Activities such as problem solving, brainstorming, identifying and locating data that has been created by others are all forms of collaboration.

As in communication, one of the most important contributions of technology to the area of collaboration is the elimination of the constraints of time and space. Face-to-face meetings are common in cases where group members are able to share the same time and same place. Telephones have removed many of the barriers of location, and voice mail has removed the barrier of time as well. In fact, experience shows that once groups have incorporated collaborative technologies into their environment, they are able to effectively minimize the number of face-to-face meetings that would otherwise be necessary to exchange information and ideas. This allows groups to optimize face-to-face meetings by limiting them to activities where this kind of interchange is most useful -- getting closure on issues, reaching consensus, etc.



Collaboration over Time and Space Variables

For the remainder of this chapter, we will focus on asynchronous collaboration (different time and place) because of its complexity and potential for improving work practices by allowing people to work more efficiently in the same time and place. Our primary goal is to define the role that shared database technology and shared information play in collaboration, starting with how the technology addresses the limits of e-mail for many-to-many interaction, and then continuing on to other forms of collaboration.

Updated: 07.11.95 22:10:06

Groupware - Communication, Collaboration, Coordination

Shared Databases: A Technology for Collaboration

In the previous chapter, we determined that messaging is a useful, general-purpose communication medium, adaptable through mailing-list capabilities to some group situations, but not really tuned to the needs of many-to-many interaction. However, the general availability and asynchronous nature of messaging has resulted in its use as a medium for collaboration. While this may appear to be a logical extension of messaging, the model falls short in many respects.

Messaging systems are primarily concerned with tracking files as messages in relation to senders and recipients. This makes it difficult for users to track information by topic. In addition, maintaining the context for discussion threads taking place over a series of e-mail messages is difficult due to the problems of tracking who responded to what and in which order. In many ways, this has prompted the extension of e-mail systems to support a shared on-line discussion space which introduces structure in the form of roles, access control, and conversational structure available to the user.

Shared database technology has evolved in an altogether different domain from messaging, its communication-centric cousin. In fact, shared databases can be traced back to the first timesharing systems found on early mainframes. Shared databases facilitate collaborative interaction by providing a *virtual common workspace* with a group-centered interface that allows participants to share information and ideas. In contrast to messaging systems which use a sending or *push* model, shared database technology supports a *pull* model for information sharing. The pull model allows users to retrieve information as needed. Users have more control over when they join various group discussions. Users are no longer held hostage to an e-mail schedule, but rather assume the responsibility to retrieve information (or to ignore it!) at their own discretion.

Shared databases also differ from messaging systems in that they not only house an entire set of messages, but also discussion items, supporting documents, arguments -- that is, knowledge -- in one place, viewable through a common structure, and providing a consistent record of what has transpired. This facilitates common understanding, and is fundamental to enabling collaborators to collectively grasp key concepts and issues. Furthermore, where a shared view is provided, multiple forms of presentation are important because individuals will want private, tailored views to support their own specialized tasks. These multiple representations may satisfy the need of a single participant to view information by date, by author, by document type, etc. They also support the need to present information to different people, for example, by customer name or market segment for the group in marketing and by part number or product name for engineering. Each form represents a different lens through which to view the collaborative task while placing information in various contexts. *For these reasons, and because work changes over time, the availability of lightweight, end user design tools is important.* They provide collaborators with the ability to customize and modify information; otherwise, the system will fall into disuse and fail.

The availability of tools, and the ease with which these applications can be developed or customized, is a key to the long term success of systems implementations. Primarily, the ability to customize user interfaces, the granularity with which data and information can be viewed and manipulated, as well as the ability to define the many variables associated with a system, are all important attributes. Commercial products that deliver a specific application, for example group conferencing, without providing access to the underlying database platform, are limited in this respect. While out-of-the-box systems like this may fill a specific short term need, they lack the flexibility and customizability needed to solve a wide variety of collaborative problems. Products that expose the underlying database platform and offer customizable tools and applications provide much more flexibility. To illustrate, we'll take the above example further. The marketing group may realize that it now needs to analyze the customer information by geographic region, in addition to market segment. To accomplish this, a new field for capturing information on region must be added to the existing database. Only a collaborative tool with access to an underlying database that allows the information in a document to be captured at the field level can be modified to easily support such specific (and changing) group requirements.

Updated: 31.10.95 21:54:58

Groupware - Communication, Collaboration, Coordination

Collaborative Applications

In developing the requirements for a flexible and customizable environment for collaboration, it is important to distinguish between the technology and the applications implemented with the technology. A flexible, shared database at the core of a groupware system provides the platform on which a wide range of applications can be developed. These may vary from simple discussion databases to rich knowledge bases that support, for example, customer assistance systems and team responses to questions.

Electronic Conferencing

Electronic conferencing systems (discussion databases, public forums) facilitate asynchronous collaboration by introducing a measure of structure that passively facilitates the process of sharing, organizing and navigating information through an interactive electronic space that serves as a common repository for contributions.

Problem-solving work, for example, comprises such general-purpose tasks as brainstorming to generate ideas, structuring those ideas, and then evaluating them. Electronic conferencing systems use shared database technology to provide the structure necessary to facilitate these steps while allowing any time/any place participation. In electronic conferencing, messages are placed in one shared database, as opposed to the individual mailboxes of a messaging system. All participants can see new messages and respond accordingly. A moderator can step in and start new topics of discussion to shift the group from one stage of the process to the next.

Leveraging Shared Databases for Other Applications

A structured database of messages can be used to provide common understanding of a discussion. Other data types can also be stored in structured form to assist groups that are working in a particular application domain. For example, in the case of group authoring, a group that is jointly writing a document should share a common draft of the document. To support their efforts to revise and comment on each other's revisions, in a database format that document can be stored as a collection of paragraphs, chapters and sections. These can be viewed in a number of ways: linearly to look like an ordinary word processing document, in outline view, in "revision mode," in old and new versions, etc. Each paragraph can serve as a "topic" for a discussion, and conversations can be structured around the controversial issues in the paper, rather than around an arbitrary set of "topic notes" typically found in an on-line forum.

This use of shared databases makes the content of any application much more useful to a group that is trying to develop material. Group-enabled applications such as word processors, spreadsheets, etc. that can store their information in databases rather than in unstructured files become much more powerful. Once that information is captured in a shared database, the information can be viewed natively through the application or through a more generic groupware interface that facilitates discussion, debate and decision making.

Updated: 02.11.95 18:48:24

Groupware - Communication, Collaboration, Coordination

Reference Publishing Systems

Reference publishing systems -- systems for publishing and widely disseminating documents -- are considered groupware because they facilitate information sharing. Information is published electronically by a provider and read by many consumers. For any given topic, this represents a one-to-many broadcast of information -- the accumulated knowledge of past experience as captured in documents such as competitive reports, forecasts and reviews, policies and procedures manuals, training materials, and newsletters and periodicals.

In order to better understand their roles in enterprise environments, it is useful to compare and contrast reference publishing with collaborative applications. Both are similar in structure -- both systems use the *pull* model to allow users to navigate and browse large quantities of public or corporate information. However, collaborative applications are interactive, whereas reference publishing applications are one-way.

In fact, reference publishing is typically so one-way that it's debatable whether to call it a collaborative application or a communication application. It is a powerful one-to-many communication tool -- with a weak distribution model. While typical implementations are one-way, support for two-way interaction can greatly enhance the usefulness of reference publishing systems.

A reference publishing system is typically a distributed database (or at least a distributed file system) containing "finished" documents. When rich distributed database technology underlies the publishing system, simply offering additional integrated communication and collaboration facilities can transform it into a dynamic environment. This environment can support learning and building of a common vision, rather than simply an information search and retrieval environment for personal use. For example, a team might use a discussion database to collaborate on the scope, direction and details of a new product strategy. The result of this collaboration is a strategy document, which in turn is stored in a reference publishing system. The document is now made available for all appropriate audiences, both internal and external. Readers (e.g., customers, financial and industry analysts) of the final document may have their own responses, recommendations and predictions. Thus the company might include a link between the strategy document in the reference publishing system and a discussion database, to which all readers (as well as the original document authors) can contribute and respond. Furthermore, contributors to the discussion database can include pointers to documents in other reference publishing systems as part of their commentary.

Updated: 02.11.95 18:49:02

Groupware - Communication, Collaboration, Coordination

The Passive Nature of Shared Databases

The model of shared databases for storing and maintaining on-line information provides many advantages over a model based on messaging. Primarily, information is pulled as needed by the consumer, thereby allowing for greater control over information *received and consumed*, as opposed to control over how information is *sent*, as in the case with messaging. However, when this technology is used without some notification or messaging support, it also suffers from limitations. Because shared databases depend on the consumer to seek out information, they become inherently passive. Missing from shared databases is a facility that easily notifies users of the addition or modification of information. For example, consider a reference library that contains standard operating procedures: as procedures are updated in the database, notification of these changes is important.

Just as an avalanche of e-mail inexorably leads to information overload, an explosion of dozens, hundreds or even thousands of shared databases eventually presents the user with an overpopulated and mind-numbing environment. A similar phenomenon occurs on the World Wide Web, with the appearance of thousands of new home pages every month. The result is that large quantities of information are stored in databases. However, important, relevant, or frequently changing information does not find its own way to the interested user.

Updated: 31.10.95 22:21:41

Groupware - Communication, Collaboration, Coordination

The World Wide Web as a Collaboration Tool

The World Wide Web (WWW, or Web) has rapidly evolved into a significant network paradigm for intra- and inter-enterprise publishing and for other collaborative applications. Fundamentally, the Web is a set of protocols which operate over the Internet (as well as private, internal networks). These protocols serve as the basis for a client/server environment that supports information sharing and, more recently, transaction processing and electronic commerce.

This rapid evolution has led many organizations to consider the Web as the basis for a much broader range of applications, many of which fall into the category of groupware as we have defined it and, most specifically, in the domain of collaboration.

The Web Infrastructure

There are three essential technologies which define the World Wide Web today because they define the communication between a Web client and a Web server connected over a TCP/IP network:

- **HTTP***, Hyper Text Transport Protocol,* governs communications between a Web client and a Web server.

- **HTML***, Hypertext Markup Language,* is the document format for Web documents or "pages." Traditional word processors use a proprietary method for representing document attributes such as font, pitch, etc. HTML uses directives instead to specify format, leaving the actual formatting to the client. HTML is a subset of SGML (Standard Generalized Markup Language), providing codes used to format hypertext linking between documents.

- **URL**, *Uniform Resource Locator,* is a pointer to a resource on the Internet. It serves as the global addressing scheme for pages stored on Web servers. The URL contains the name of the Web server (e.g., WWW.Lotus.Com) and possible extensions that define a specific Web page or other information to be used by the Web server.

While many people today think of the Web through a lens focused on products, there is no doubt that this will rapidly change, and that we will view the Web as a set of capabilities implemented through specific protocols that are supported by a wide range of products. Although Web browsers as we know them today will continue to be used, we can fully expect Web browsing functions to be directly incorporated into other programs.

A "Web browser" will be any program that implements Web client protocols, and a "Web server" will be any program that implements Web server protocols. Soon, we will be in an environment where applications such as mail systems use Web protocols to access database servers that also happen to implement Web protocols. Hence, it will be the protocols that both determine and limit what can be done on the Web. The evolution of Web protocols (specifically HTTP and HTML) has matured through two phases, and is now entering a third phase.

Phase One -- Information Publishing. The first version of HTML supported only information publishing through a hypertext model. A document segment could point to other document segments in different documents on different servers so that users could move through an information "space" in a non-linear fashion.

Anyone could create Web pages in HTML and store these on an HTTP server, and anyone with a browser could use HTTP to access the Web page. Web developers quickly realized that there were tasks that could not be accomplished with the simple Web protocols, so a "trap door" or application exit was developed called the common gateway interface, or CGI. A URL, in addition to referencing a specific page on a Web server, could also reference a program to be invoked on the Web server through the CGI. One thing a CGI program can do is dynamically create Web pages. This opened a whole new world for the Web because "dynamic publishing" could be supported in addition to classic electronic publishing. Let's consider two simple examples to see the difference.

In the first case, let's consider a Web server that stores research papers and makes them electronically available to browsers worldwide. Once the papers are stored on the server, they don't change. An index page may be rewritten every time papers are added to the server, but the papers themselves do not change. It is a static Web site.

Now let's consider a firm that carefully tracks companies in a specific industry (e.g., airlines) and wants to make this information available to subscribers on the Web. Some of the information is static and, once defined, can be stored in standard Web pages and accessed via Web links. But the publisher might also want to make real-time news a part of the service. Information about a specific airline might include standard corporate overviews, but also late-breaking news. If a subscriber selects "news" from one of the predefined pages, this might invoke a CGI program that accesses a news feed service, gathers up information about that airline, formats a Web page and transmits the Web page back to the browser via HTTP. This Web site is supporting dynamic publishing.

Much of the high value interaction on the Web comes from pages dynamically created via CGI programs. Hence, the gating factor in building this class of applications is not what is in HTML and HTTP, but rather the kind of application development environment that is available for writing CGI programs that interact with external data sources. This is why we are now beginning to see a new generation of applications development tools for the Web. It stands to reason that rich application development environments that already support collaborative applications will also support Web protocols.

Phase Two -- Fielded Forms. Let's take the airline example a step further. Real users would probably like to be fairly specific about the news items they are seeking (e.g., news on the airline for just the past 24 hours, or just articles that contain the words "airfare" and "international"). We've now moved from dynamic publishing to *interactive queries*. In order to support this requirement, we need to present a page to the user in which specific search terms can be entered, for example. The extension of HTML to support "fielded forms" defined the second major step in the evolution of the Web. Once data is entered into a field, a URL is sent back to the server that invokes a CGI program that extracts the data from the form, processes the information, and dynamically creates a page and sends it back to the user.

Not only do fielded forms enable the Web to support interactive query applications, they also enable electronic commerce and transaction services (e.g., airline reservations systems, stock quoting services, retail catalogs and order forms, customer support systems). The advantages of Web-based transaction processing to the supplier are (1) worldwide connectivity through the Internet and (2) universal support for any client machine that has a Web browser. While the Web may have started as a collaborative system for information sharing via a publishing model, the excitement around the Web today is very much about transaction systems and electronic commerce -- domains outside of what is normally considered collaborative computing. As we move toward this model of dynamic publishing, programming beyond HTML via CGI is required, which leads to the next phase of evolution of the World Wide Web.

Phase Three -- Programming Languages. The initial version of HTML specified how a document should be formatted. Examined through a slightly different lens, however, HTML was really a simple programming language. An HTML statement could, for example, state that "when a user clicks the mouse on this hot spot, send the following hot spot coordinates as a request to the appropriate Web server, to which the Web server returns a URL." This is a simple program. It is the evolution of HTML that will define the third phase of evolution for the Web.

Just as programming languages have evolved toward an object-oriented model to support the demands of modern applications, HTML will also evolve toward an object-oriented model. Browsers can include code that can interpret a program, and Web pages can include programs. When a Web page is accessed and transmitted to a requesting browser, the browser interprets the program and executes the requested action (e.g., providing animation to an image). The emergence of Web-specific languages dramatically extends the programmability and the extensibility of HTML and, therefore, of browsers.

Constraints of the Web

The Web is evolving rapidly, and its boundaries and constraints are not easily discernible this early in its lifecycle. Over time we will learn that, like every innovation, there are boundaries that must be understood and respected. It is already clear where we will begin to see some challenges in the near future.

What initially made the Web and Web browsers so attractive was the utter simplicity of the technology and the resulting product. Web browsers were compact, easy to implement and easy to use. Complexity will increase as new browsers grow in functionality, including:

- Support for multiple coding formats (e.g., Adobe Acrobat in addition to HTML).
- Inclusion and support of multiple languages.
- Support for a local file system (e.g., a local message store for a mail system written for the Web), and its use for caching Web pages or storing replicas of data on Web servers.
- The addition of rich security features to browsers and better access control on servers.

To be clear, this is not a gloom and doom forecast for the Web. To the contrary, the Web has had and will have a profound effect on the information technology industry and on society. However, by adding functionality to the Web we necessarily add complexity.

The introduction of complexity, in turn, makes interoperability more difficult. The Web today is truly open. Web servers and Web browsers are interoperable because the protocols are simple and uniformly implemented. Now that the Web has evolved from a research project where concepts like profit and market share were non-issues to a burgeoning market-driven industry, we are beginning to see *proprietary* extensions to HTML, so that a particular vendor's Web server product works best with *that* vendor's browser. This may have the negative effect of requiring users to implement multiple browsers, perhaps using browser A because it works best with the user's stock quoting service and browser B because it is optimized to work with the user's local department store server. We will see complexity ratchet up yet further until the inevitable "standards" wars are waged and resolved.

We will continue to see dramatic investment in the Web and associated technologies. Web technology has evolved from collaboration (dynamic information publishing) toward electronic commerce. While there will continue to be innovation in the collaborative aspects of the Web, it's likely that the center of gravity will move more toward electronic commerce. Groupware, also starting from a collaborative base, will see its center of gravity move toward coordination applications, the subject of the next chapter. Because of their respective evolutionary paths, we can expect to see far more complementarity than overlap

Updated: 02.11.95 19:20:54

Groupware - Communication, Collaboration, Coordination

Conclusions

In examining the mix of communication and collaboration-based technologies the following is clear:

- The requirements of collaboration and communication are distinct. Therefore, it follows that electronic messaging alone does not sufficiently facilitate the process of collaboration. Database technology employs a "pull" model of information distribution that engages users in the collaborative process.

- Collaboration requires a system that combines these *push* and *pull* models, and provides a robust framework for exploiting the many ways that users need to *communicate* and *collaborate.*

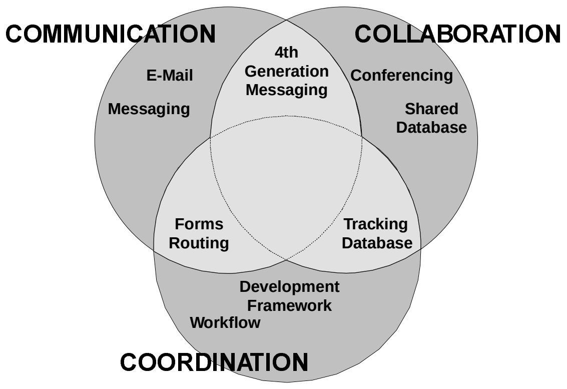
- A shared database is essential for common work, shared views and for crystallizing information into organizational knowledge. One way to leverage the integration of the push and pull models is through tools that support a coordinated use of messaging and shared database technologies. These mirror the need for groups to coordinate their work efforts -- both in the sense of sequencing their activities and smoothing the transitions between different modes of work.

The third category of group work -- *coordination* -- is supported by both of these technologies, as well as by tools that allow groups to "program" their combined use of the two. That is the subject of the next chapter.

Updated: 31.10.95 22:34:18

Groupware - Communication, Collaboration, Coordination

Coordination



[<img src=
"circle3.gif">]

Thus far, we have discussed how groups of people communicate and collaborate in order to share information and leverage knowledge that helps them perform their jobs more efficiently and more effectively. What characterizes much of this interaction is its *ad hoc* and unstructured nature. That is, people send each other e-mail messages at their own discretion, and they refer to shared resources when the need exists. The activities occur on an as-needed, dynamic basis. When we think of collaboration, we think of "brainstorming" sessions, co-authoring a research paper, or other "we don't really know where it's going to end up" sorts of creative activities.

However, many business activities are much more structured in nature. Enterprises do not expect people to "collaborate" on processing an expense report; rather, the enterprise defines specific policies about how an expense report is to be routed through an organization so that it is properly approved, is auditable and is secure. Many people are involved, but the enterprise's policies specify, or even dictate, the *coordination* required between these people to meet a defined objective. The successful completion of a pre-defined business process depends on the coordination of people in completing a set of structured tasks in a particular sequence and within expected time constraints. To a great extent this has been the domain of workflow automation systems -- a focus on highly structured business processes that exhibit *pre-defined, conditional* workflows based on status and conditions. Whereas collaboration is relatively passive from a systems perspective (we create a common workspace but we do not dictate how the space is used), coordination is very active from a systems perspective (we specify how activities are to be accomplished).

When we choose to move from collaboration to coordination for a specific problem, we implement workflow systems in which we define forms, specify operations on these forms, specify routing logic for the forms, specify how external data is to be accessed or modified, specify triggering actions that occur when certain conditions are met, etc. We develop *workflow applications*, and to do this we need applications development tools. Coordination, as used here, refers to the use of application development tools for a class of applications generically referred to as workflow, where a major attribute is often "tracking" some resource. The essential tool for building coordination systems is an applications development environment.

While exploiting structure in workflow applications is important, it overlooks a significant segment of coordination which relates to tasks and activities that are not pre-defined. In fact, most real work involves a combination of highly structured processes and tasks where the process is fuzzy and the rules, routes and roles are dynamically defined as the work is being done. This is why workflow systems alone, without the collaborative and communication components that provide for "soft" interaction, are often unsuccessful and deemed to be "too rigid."

For example, consider a software bug fix application. The structured steps might include the initial registration of the bug, submission to a project manager, assignment to a programmer/analyst, routing to quality assurance, delivery to a configuration management specialist, and posting to a public reference library (e.g., the World Wide Web or bulletin board) for downloading to customers. Throughout this process, however, there are likely to be several unstructured steps that cannot be anticipated or automated, such as referring to a trouble tracking database for help in identifying similar bugs and fixes, and e-mail requests for more information from various parties.

Coordination, then, is more than the automation of a sequence of structured tasks, bringing people into and out of a process as needed. Rather, when we look at how work is really done, we see that knowledge that is essential to the completion of a process is acquired as a result of the relationships among the various participants, outside of the context of the process itself. Complete coordination includes support for informal conversations (through e-mail, discussion databases and reference publishing systems) that allow people to gather the information they need to get their jobs done, especially when these conversations happen in the context of a more structured process.

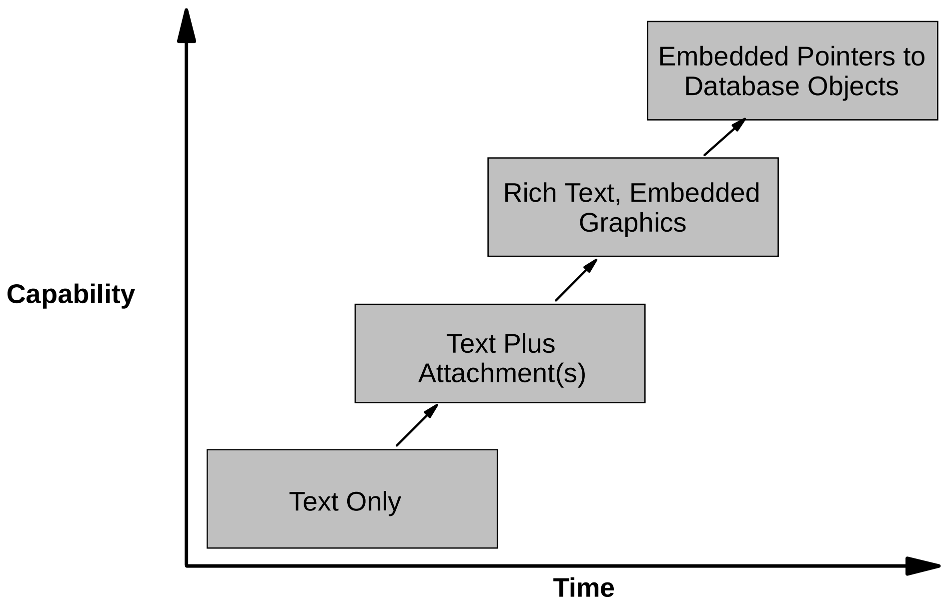
In previous chapters, we examined how the individual models of communication and collaboration, based on pushing and pulling information, each suffered from incompleteness when applied beyond its original design point. An integrated solution that *coordinates* the use of both messaging for notification and shared databases for collaboration provides a more balanced and comprehensive approach to supporting structured and unstructured processes.

Updated: 07.11.95 22:10:07

Groupware - Communication, Collaboration, Coordination

Four Generations of Messaging

The fourth generation of messaging systems exploits the coordinated use of messaging with shared databases. When we look back at how messaging systems have evolved we can see that first generation systems were only capable of supporting simple text messages. Second generation systems augmented this with the capability to attach binary documents to simple text messages. Third generation systems provided support for rich text (i.e., color, multiple fonts, character sizes, etc.) and embedded objects in the message body itself, as well as in binary attachments. And finally, fourth generation systems represent a significant advance in messaging through the support for hypertext links to documents in shared databases and file systems. Rather than attaching an object to a message, we include a "doclink" -- an electronic pointer to the object in a shared database -- in the message. When a user double-clicks on the doclink, the object is automatically and immediately retrieved and presented to the user. Messaging is improved because it no longer requires that users include objects and attachments in messages. Shared databases are improved because users now have a means of notifying others of the existence of relevant or important information that otherwise might have languished in a database, unused and unnoticed. We refer to this as *integrated messaging and groupware*.

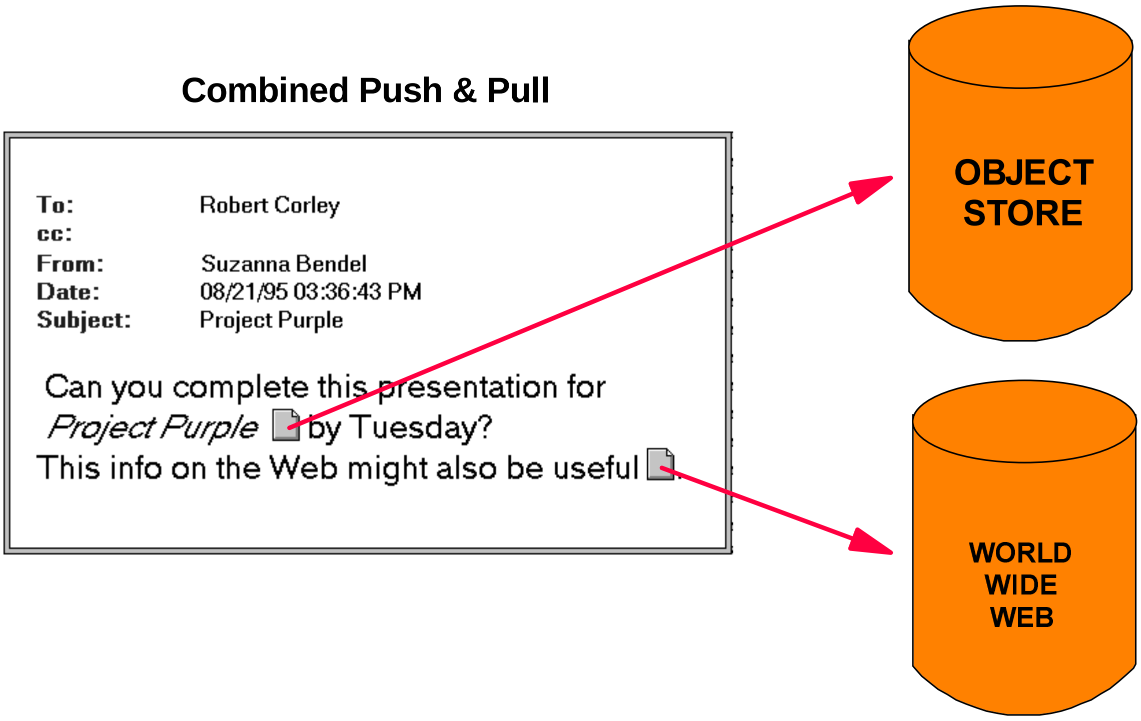

Embedded Pointers to Database Objects
Rich Text, Embedded Graphics
Capability
Text Plus Attachment(s)
Text Only
Time

Four Generations of Messaging

Consider a coordinated activity in which an advertising director regularly sends out drafts of print ads, complete with text and graphics, to a group of editors for review. For each ad, the director uses e-mail to distribute the ad to the review group. The ad is sent as a binary attachment to the e-mail message. Responses from various group members -- in the form of actual changes to the ad or suggestions and comments -- quickly result in a version control and document management problem. At the same time, pushing large messages back and forth dramatically increases the volume of messaging, and thereby, network traffic.

Alternatively, using a shared database, the director avoids many of these problems. The shared database allows the director and reviewers to keep track of the most recent version of the document as well as the threaded discussion that led to various changes. On the other hand, this solution lacks the notification capability needed when a new ad has been placed in the database for review.

Combined Push & Pull


To: Robert Corley
cc:
From: Suzanna Bendel
Date: 08/21/95 03:36:43 PM
Subject: Project Purple
Can you complete this presentation for *Project Purple* by Tuesday?
This info on the Web might also be useful.
OBJECT STORE
WORLD WIDE WEB

Fourth Generation Messaging

The coordinated use of messaging and shared databases in a fourth generation e-mail system (i.e., integrated messaging and groupware) resolves both of these problems. When the director posts a new ad to the shared database, an e-mail message containing a hypertext link or "pointer" to the document is also created. E-mail is used to notify reviewers of the new ad, but, the new ad is not physically transferred as an attachment to the mail message. Instead, the document link in the mail message dynamically transfers the reviewers to the new ads which remain resident in a shared database. This allows the ads to be managed and maintained centrally, providing version control and support for collaboration in a shared space.

Updated: 02.11.95 22:04:28

Groupware - Communication, Collaboration, Coordination

Integrated Messaging and Groupware

In an earlier chapter, we noted that e-mail users are likely to think of groupware in terms of messaging, while users of conferencing and on-line publishing systems see groupware as a function of shared databases, and users focused on the automation of structured business processes are inclined to perceive groupware as workflow automation. In fact, each of these technologies on its own does represent a specific dimension of groupware. Messaging and shared database technologies have each become the foundation for workflow automation systems. Messaging has moved into the workflow space by exposing its APIs to application development facilities and tools to create a workflow routing approach to automating processes. Shared databases have been similarly extended to support a tracking approach to workflow automation.

Messaging Model of Workflow

Workflow automation is typically associated with the automatic routing of documents such as expense reports. Route-based workflow automation generally uses the underlying messaging system to route documents to the next person who must take an action (e.g., approve the expense report). The route can be hard-coded, or a rule may determine the routing path based on a specific value (e.g., the amount of the expense) or on a person's role (e.g., the initiator's supervisor). These rules can be sophisticated and may even be able to call an external application to retrieve some data (e.g., a supervisor's authorization limit).

Routing-based workflow is powerful because it matches the model of routing paper: the document is acted upon and sent to the next person for further action.

There is a significant drawback to the routing-based workflow model: as the document is being routed, it becomes unavailable to anyone other than the person in whose inbox it now resides. The problems this can create are illustrated in the following contract routing example, in which a contract is being negotiated with a customer. In order to be signed, the contract must be approved internally by several people. A mail-based workflow system routes the contract to each person requesting approval, denial, and/or comments. Suppose the customer would like to know the status of the contract, or would like to make changes to the contract during the approval process. Is it possible to determine who has the contract at any given point in the process? If so, is it possible to retrieve it? At which point can changes be introduced? What happens to the approval process once a change has been implemented?

Anticipating some of these problems is possible (perhaps rewriting the rule or adding a new rule that accounts for when a person goes on vacation), but building strategic applications on patches such as this is uncomfortable for companies, and anticipating all conditions and exceptions is impossible.

Shared Database Model of Workflow

The second workflow model is the shared database. In this model, users consult a tracking database to check the status of specific documents.

The shared database model has three advantages. First, the database sits on a server and is subject to server-based processes (such as RDBMS triggers, agents or macros) that can initiate action without any specific user activity. In many cases, the action may be the direct result of a lack of user activity (a sales person has not contacted a customer in 30 days, a monthly report has not been submitted, a contract to be approved is waiting for a specific person for over 24 hours, etc.) or an external condition (inventory has dropped to the reorder point, a client's credit rating has changed, a deadline is approaching).

Second, the shared database model keeps the document or record in question available for others while the workflow proceeds. In the contract approval example, the changes could be made to the original document in the database, and, depending on the changes, the workflow could continue or be aborted and launched again.

The third advantage is that the shared database model makes the management and macro-management of the workflow much easier. The server can both monitor specific instances of the process and keep aggregate statistics about the overall process, the latter allowing better management and planning of the workflow.

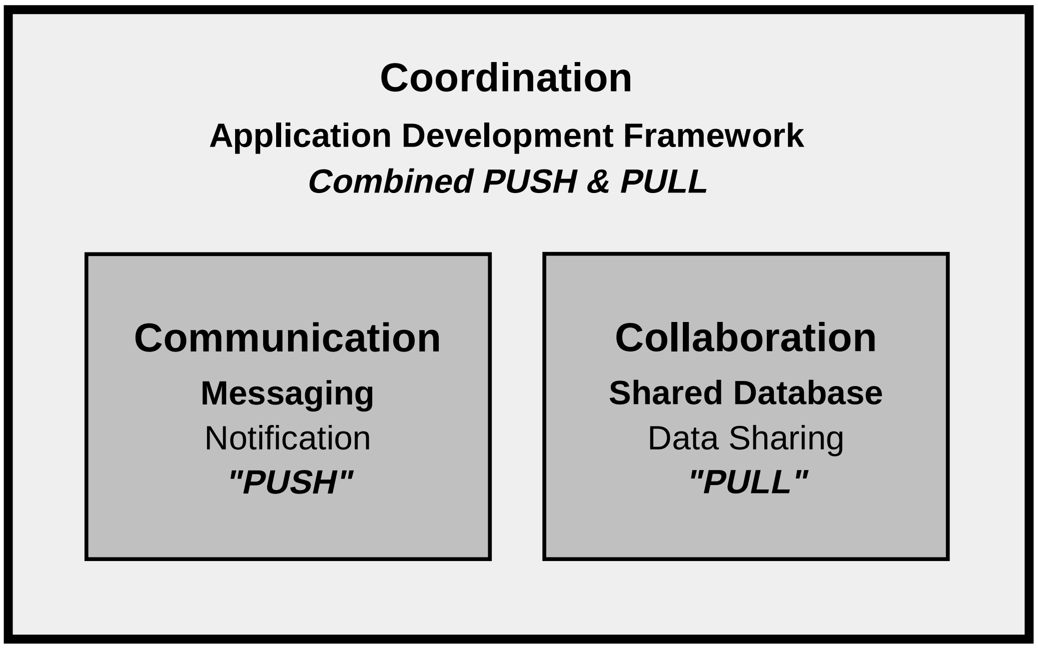
The primary constraint of the shared database model is the lack of event-driven notification by the system to the workflow participants. That is, it is incumbent upon the user to check the database.

An Integrated Model

These constraints of the messaging and shared database models of workflow sound familiar because they are reflections of the constraints of their underlying technologies. Messaging is very efficient at sending a document, but provides no way to manage the document as it proceeds through its route. Shared databases are proficient in managing documents and providing an overview, but are poor at alerting users of a change in state or information.

We have already seen the advantages of a fully integrated messaging/shared database system through fourth generation messaging. If we generalize this model, it becomes obvious that workflow applications built on a common platform that natively supports both models -- a messaging subsystem with conditional routing capabilities and a shared database for storing, retrieving, viewing, and managing work processes -- provide the robustness and flexibility needed to effectively automate work processes.

To illustrate, let us look at how the same contract routing and tracking system discussed earlier might be implemented using an integrated model. The contract is created and stored in a shared database. When it is saved, a mail message goes to the first approver. The message is not the contract itself, nor does it tell the approver where to find the contract. The message contains a hypertext link to the contract, which, when activated, will launch the contract for the approver's use while maintaining it in the shared database. Further approvals and routing can be done, but the most up-to-date version of the contract itself is always available in the database. Any requests about status can be answered by anyone with access to the database.



Groupware Building Blocks

Updated: 02.11.95 21:13:40

The Extended Transaction Model

As we have seen, there are two basic approaches to workflow systems, one based on a messaging model and one based on a shared database model. Workflow systems that support both models, as well as supporting the less structured collaborative functions, will be most successful.

It's important to recognize, however, that these workflow systems rarely exist in isolation. More often than not, the workflow processes are the "front-office" components linked to specific "back-offfice" systems that have long been automated. Hence, the linkage between the "output" of the workflow system and the "input" of the more classic transaction processing system is critical. An example will be helpful, and the purchasing process serves as a good example.

A typical "purchasing system" begins with an approved purchase order requisition, and produces an official purchase order. These systems are typically very well automated, and most of the cost reductions to be gained from automation have been realized. The input to the system is an approved purchase order requisition. However, an approved purchase order requisition is really the result of a workflow process that begins when someone decides that he or she needs to purchase something. That individual creates a purchase order requisition, and this requisition "threads its way" through the system until it is finally approved or rejected. When it is approved, the "transaction" in the classic sense is initiated. In reality, however, the business transaction was initiated when someone created a purchase order requisition. Almost all back-office systems have a significant front-office workflow component.

We refer to this broader and more business oriented view of a transaction as the *extended transaction model.* By tightly linking the front-office workflow process to the existing back-office process, we can achieve significant economic gains. To achieve this tight linkage, we need to exploit tools that allow us to interact with the existing transaction systems in a very controlled way. Many transaction processing systems support this form of programmatic interface.

By viewing workflow as part of an extended transaction, it becomes extremely clear that the key to developing rich workflow/coordination applications is an application development environment that is integrated with and part of the workflow system.

Updated: 02.11.95 22:14:30

Groupware - Communication, Collaboration, Coordination

An Application Development Framework

There are a number of essential components to a rich application development environment for building coordination applications:

- A rich forms designer and filler tool is required. Forms will include text, images, sound, fields, buttons, list boxes, etc. and may consist of subforms. These subforms may be windows or dialog boxes with familiar list boxes, combo boxes, etc. Strong editing capabilities are required for fields in the form, and this editing may require interfacing with external databases to validate user entries.

- Some form of programming capability is required, in the form of a scripting language, macro language or full programming language. A rich set of APIs to the underlying technology is required. Developers must be able to provide instructions (through scripts, for example) whenever events such as opening a form, changing a field or adding a new document to the database occur.

- An agent capability is required so that actions can be taken (such as a script being run) whenever a specified event occurs.

- A development environment that supports a rich debugging capability is required.

- End user definable views are required to support the customized display of information based on task, individual and group requirements.

When these tools are properly applied to develop workflow applications, the result can be very robust applications that exhibit a natural user interface and at the same time have the control necessary for enterprise applications.

Updated: 02.11.95 22:16:43

Groupware - Communication, Collaboration, Coordination

Conclusions

In most business processes, the process is fuzzy and the rules, routes and roles are actually determined as the work is being done. It is in this fuzzy aspect of work where leveraging knowledge is most critical, and where business professionals are responsible for managing their jobs. Our discussion of the structured and unstructured activities that comprise every business process yields the following conclusions:

- Most real work involves dynamic movement between structured, unstructured, *ad hoc*, and predefined work, requiring an integrated push/pull model to support users as they move from one type of work to the next in the normal course of a process.

- Structured, pre-defined group activities can be supported by programmatic workflow applications, of which there are two basic types: routing, based on messaging technology; and tracking, based on shared database technology. The integration of these two approaches to workflow automation is achieved through an integrated application development framework which exploits services of both messaging and shared database systems.

- The application development environment is a key component of a groupware system architecture. This is discussed more fully in the chapter "Architectural Considerations."

Updated: 02.11.95 22:20:54

Groupware - Communication, Collaboration, Coordination

Architectural Considerations

So far, we have established the need for a groupware infrastructure that not only supports applications that individually depend on communication, collaboration or coordination, but exploits the synergy created by the integration of all three.

From these reference points, we can determine the key infrastructure requirements.

- An **information model** based on a object store/distributed shared database that houses and manages data, regardless of original source.
- A **distribution and access model** based on messaging and database replication for the movement of data to and from anywhere and anyone in the organization.
- An **application development framework** that leverages the native underlying services of the object store and distribution/access model for the development of custom groupware applications.

While these represent the core components of an integrated groupware architecture, integration with external data sources, security and directory services are also critically important.

Updated: 02.11.95 23:14:02

Groupware - Communication, Collaboration, Coordination

Information Model: The Groupware Object Store

The object store is the heart of a groupware infrastructure. In order to effectively become an enterprise information repository, an object store must be defined by its ability to provide shared access among users and applications as defined throughout this document. Specifically, the object store is the message store for communication applications, a virtual common workspace for collaborative applications, and a shared database for coordination.

The object store should be internally consistent across all of these applications. In other words, there is no reason the internal structure of a message store for a user's inbox should be architecturally different than that of a conferencing database or of a workflow application. The *appearance* at the user interface level may be significantly different, but at the deepest level the organization of the data into individual fields, rich data types, attachments and objects can, and should, be consistent.

This model provides the following benefits:

- Consistent method of handling information throughout all stages of communication, collaboration and coordination -- for end users and application developers alike.
- True separation between data and applications.
- One consistent set of information for many uses -- eliminates data redundancy and the problems associated with it.

The groupware object store is a distributed, shared database. At its lowest level the object store holds objects. A distributed groupware environment consists of an arbitrary network of servers. Within a server, there is a set of individual databases. Within the given database, there is a set of *documents*. Within a document, there is a set of *fields*. A document is normally how data is presented to the end user. This is the basic unit of storage in the object store. The ability to manage information at this level of granularity allows significantly more powerful information handling than the traditional message store allows.

To be a sustaining information management system, the object store should support the following:

- **Rich objects.** The breadth of object types that an object store is capable of supporting serve to define or limit the variety of applications that it can support. Documents should support a wide variety of objects including numbers, text, rich text, graphics, images, voice, video, links to other documents, and embedded applications.
- **Document Hierarchy.** An object store requires a facility for document threading through preservation of the parent-child relationship between documents and the responses to them. This is most prevalent in group conferencing and discussion databases.
- **Versioning.** Support for versioning when changes are made to documents is critical for document sharing where multiple authors are involved.
- **Hypertext Links.** Support for links between documents within and across different databases provides for the greatest level of flexibility in referencing information contained in databases. These links should be a part of the document so that they can be maintained and managed in a distributed environment.
- **Consistency.** All applications of the underlying object store should be based on a fundamental set of architectural principles. This consistency enables a number of important features and capabilities of the system as a whole. Full text indexing and retrieval capabilities embodied in the groupware system can uniformly act on all types of storage, be it e-mail or a help desk application. This enables a consistent user experience for data and applications, regardless of form or location.

Updated: 02.11.95 23:17:03

Groupware - Communication, Collaboration, Coordination

Distribution Technologies

Store-and-Forward Routing

Store-and-forward routing is key to messaging and the push model of communication. The distribution model for messaging is based on a store-and-forward asynchronous transport. This transport is responsible for routing information from senders to recipients as characterized by the *push* model associated with communication. SMTP/MIME and X.400 have emerged as the industry standards for messaging transports. Support of these standards is important as they play an important role in supporting messaging interoperability between heterogeneous messaging environments within and between enterprises.

Once the messages have reached their ultimate destination (i.e., the message/object store), a user can replicate the messages from the server to the client for use in a disconnected mode. In this way, from a document perspective mail messages are treated like any other document in the object store, inheriting all the services of that object store (see Client Replication in the following section). For server to server routing, however, mail messages are distributed differently from other documents through store-and-forward mechanisms.

Replication

The groupware object store is a distributed, shared database. While this is in keeping with the benefits that a distributed computing model offers, technology is required to present a consistent and logical view of physically distributed information. Database replication accomplishes this by synchronizing changes to multiple copies of the same database at geographically dispersed sites. For example, a remote site in San Francisco can make a replica of a database in Paris. This allows users in San Francisco to access this information on a local server as opposed to connecting to the database server in Paris. Replication automatically synchronizes changes made in both locations so that workgroups in San Francisco and Paris have a consistent logical view of the database. The process of replication involves examining documents that have been added, modified, or deleted from each server and then updating both databases so that each is identical to the other.

While the process of replication may appear to be straightforward, it is important to examine replication mechanisms more closely when selecting technology. Implementations vary widely, having a significant impact on network topologies required to support a distributed environment. The following is a closer look at the requirements of replication:

- **Bi-directional Replication.** Once two replicas of a database are synchronized, workgroup members on different networks and servers begin to make changes, deletions and additions to them. That is, as soon as the replication process is complete, the two replicas begin to fall out of synch with each other. Therefore, at the next replication, the server in San Francisco should be able to replicate all changes, additions and deletions to the server in Paris, while at the same time replicating back all the changes, deletions and additions made on the Paris server. The replicator should flag any conflicts.

- **Efficiency.** Given that networks vary widely across an enterprise, the replication process should be highly adaptive and optimized to minimize the volume of network traffic. The most important determinant for network utilization is the granularity of replication. For example, the server in Paris does not need to copy the *entire* San Francisco database each time there is a change at the field or document level. Only those fields which have changed in either location need to be replicated.

- **Client Replication.** Mobile or nomadic users in a workgroup need the same level of access to server databases as connected users. The way that a user accesses and works with information should be consistent regardless of whether they are connected to the network or not. Therefore, replication should not be limited to server-to-server connectivity, but should also include client-to-server connectivity. Client-to-server replication gives the mobile user the capability to maintain a local replica of a database (or several different databases on several different servers) and work with it off-line in exactly the same manner as when connected to the server. Once the user connects to the server again, replication is used to synchronize changes that were made off-line. At this point, server-to-server replication can take over to distribute the changes across the enterprise. The ability to select documents and databases to take on the road is a natural extension of the way we pack briefcases when traveling on business. Since anything that can be stored in a database can be replicated, client replication gives users the ability to maintain a consistent level of communication, collaboration and coordination any time, any place. Nomadic users can take the message store (e-mail), discussion databases, reference information, and any coordination or workflow application on the road using a single method.

- **Programmable Control.** Replication should give a great degree of control to administrators and end users by allowing them to selectively replicate documents based on relevant criteria; for example, documents that have changed as of a certain date, documents by author, size, or customer name. Given time and resource constraints -- such as limited access to expensive telephone lines, wireless communications or limited disk space -- mobile users have an even greater need for this kind of programmability.

- **Application Distribution.** One of the challenges of managing a cross platform distributed client/server environment is the installation and upgrade of applications. This is further exacerbated in a groupware environment where group needs are constantly changing. For these reasons, an application deployment environment should leverage the data distribution/replication capabilities of the object store. *That is, replication should be used to distribute application objects and changes to applications just as easily as data.* Each time a user accesses the object store, the most current version of the application is launched. The groupware infrastructure uses replication to distribute and deploy data and application updates automatically through client connection to a server. This can save application developers and systems administrators the tremendous burden of manually installing new or modified applications to distributed clients and servers across an enterprise.

- **Point-to-Point Transfer*.*** Reliable synchronization of databases requires an interactive means of "handshaking" between systems. By definition, the model for this must be synchronous. RPC (Remote Procedure Call) provides a reliable transport for point-to-point replication. This is accomplished through direct connections with the end point for replication.

  This is in contrast to the asynchronous model used in the messaging store-and-forward transport which routes messages through many intermediate hops. In this model, changes that are made to the database would be mailed to other databases. At first glance, this appears to be a clever solution, leveraging an existing messaging infrastructure. However, in practice store-and-forward replication is distinctly complex and difficult to manage. One of the most intractable problems is that guaranteed delivery of message data is much less deterministic than in a point-to-point environment. Store-and-forward messaging implies, for example, an unknown periodicity; in other words, one cannot necessarily predict how long an electronic mail message may take when traveling across several hops. This is particularly true in large corporate and public networks in which the route of an e-mail is determined dynamically, that is, as the message is sent (and hence may be different each time). Thus it is difficult to know when, or even if, the object stores are in fact up to date.

Updated: 02.11.95 23:25:47

Groupware - Communication, Collaboration, Coordination

Application Development Environment

An important characteristic of a groupware platform is its ability to support rapid application development and deployment. The platform should be capable of supporting a full spectrum of application development, ranging from end users with no programming experience to power users to professional developers. Clearly, each set of developers has their own set of requirements, with end users depending on ease of use and professional developers relying on the robustness of the development environment.

A robust development environment should effectively shield the developer from most application deployment considerations.

- **Platforms.** Because the groupware infrastructure acts as a layer between applications and the underlying hardware/software operating environments, applications can be developed without thought to their eventual desktop, network and server operating systems. All system-specific attributes of an application, such as the individual characteristics of a desktop graphical user interface, are supported without any additional programming or modifications.

- **Application Distribution.** Because application logic is handled like any other object within the distributed object store, applications developers can leverage the distribution capabilities of the object store for application distribution. Developers need not concern themselves with distributing and installing applications and updates. The applications are self-distributing, and can be deployed easily to any site, regardless of geographical location.

- **Mobile Support.** The ability of a groupware infrastructure to support mobile users frees developers from creating customized mobile versions of their applications. Any application behaves the same regardless of its use in connected or disconnected mode.

Programming Languages and Tools

As already discussed, in an integrated messaging and groupware infrastructure, application logic is within the object store. In addition to relieving the developer of deployment issues, it provides a modern platform for rapid application prototyping. Once logic has been added to a database, no compiling is required. The application can be distributed to a set of users for testing and acceptance. Any changes requested by users can similarly be made and redeployed without a lengthy redevelopment and redeployment process.

An integrated programming language within this environment should meet the following two sets of criteria.

First, the programming language must meet the basic requirements of any professional development environment. This set of requirements distinguishes it from development using templates and macro languages:

- **Fully Structured.** To create sophisticated groupware applications, developers must have complex logical and flow-of-control capabilities, as well as programming constructs such as looping and branching in the programming language.

- **Professional Development Environment.** Unlike the bare bones command line interface that distinguishes most product macro languages, a robust groupware programming language, like any programming language, requires a modern, full-featured interactive development environment, including a sophisticated editor, a class library browser, and interactive debugger. This is a reflection of the fact that a programming language is a tool for professional developers and power users with real programming experience.

- **High Level of Abstraction.** Modern client/server systems and their event-driven graphical user interfaces demand a high-level of skill on the part of developers.

The language should allow the developer to create graphical, event-driven interfaces without having to resort to low-level programming.

- **Modern UI Support.** The programming language must facilitate development of user interfaces in a modern GUI environment. It must be easy to create, display and edit forms, create pop-up windows (e.g., dialog boxes), and support standard GUI interface tools such as buttons, list boxes, etc. Strong event handling must be supported so that a developer can specify what action should be taken when a user clicks on the mouse or enters data into a field, for example.

Second, a well-integrated language should take advantage of all the groupware platform system-level services. This serves to distinguish between development tools that *support* groupware application development but do not necessarily *exploit* the platform services unique to a groupware infrastructure.

- **Support for Client *and* Server Platforms.** Most groupware applications include functionality on the server as well as the client. Applications created with a programming language, therefore, should not be constrained to client-only or server-only scripts. This is in contrast to traditional database scripting languages, which operate only on the server as stored procedures. Likewise, desktop programming languages have naturally operated only on desktop platforms. A single groupware programming language should run across both sides of the network, taking full advantage of the client/server architecture.

- **Native Access to System Services.** Developers require complete access to the capabilities of the object store and messaging infrastructure. Facilities such as replication, security and messaging must be natively accessible to the developer using the programming language.

- **Multiple Platform Support.** Modern client/server groupware applications support entire enterprises, and often cross company boundaries to include customers, suppliers and business partners. Programs written with a groupware programming language, then, must also run on the full complement of client and server platforms. Moreover, these programs should run without change or recompilation due to platform-specific requirements.

- **Interoperability.** It is likely that professional developers will employ more than one programming tool in order to create a single application or to modify an existing one. That is, developers might also use the templates, pre-programmed buttons, and macro language of a groupware environment. Therefore, the programming language should be fully interoperable with these other development objects. In addition, the language must allow developers to freely call third-party APIs to forge complex system integration.

An integrated programming language that meets these criteria is an attractive alternative to templates, macro languages, and third-party and stand-alone tools and languages, in creating high value client/server groupware applications.

End User Development

The greatest challenge to a groupware infrastructure is the need to reconcile two diametrically opposed principles: creating applications for use by groups, while at the same time accounting for the likelihood that individual users will want or need to view components of that shared application in a completely unique fashion.

In discussing the essential characteristics of a groupware application development framework, we mentioned the need for native access to the services of the groupware object store and distribution services. Similarly, users themselves need native access to the data stored in the object model in order to manipulate it toward specific ends. Examples of user-defined extensions to existing applications include:

- **Customized Views.** There is no single, optimal format for presenting information. Each member of the group will have his or her own special needs, with some preferring documents ordered by date, others by name, and others by title. Furthermore, the amount of information presented on the screen is also a matter of preference. For this reason, users will modify an existing application by creating their own private views of information.

- **Categorization.** Most people organize information by breaking it down into categories. A category might be a project name, a customer reference, or a meaningful topic (e.g., competitive information). In fact, some items might fall into more than one category. The important point here, of course, is that it is the user who defines these categories, since it is the user to whom the category names hold meaning. That is, all documents that relate to a bug fix project might be categorized by one person under the heading "bug fix," by another person under the heading "Top Priority," and by yet another person under the project's code name.

- **Agents.** Every person has their own perception of what is important, relevant or urgent. Receiving information from others who decide on their own what is urgent or important is one thing. Another thing entirely is one's ability to sift through a mountain of information, deciding for oneself what is or is not important. By building agents, individual users can automate the search for information so that the groupware system itself seeks out, finds and retrieves information based upon a set of user-defined criteria.

- **Broadened Scope.** Applications designed for relatively small groups frequently grow to include more users. The marketing department includes members from sales and product development in a business process, and needs to provide those new participants with access to an application and the data it contains. Rather than going back to the original designer, qualified users are able to make changes to the access list, broadening the scope of the application beyond its original group.

Updated: 02.11.95 23:30:33

Groupware - Communication, Collaboration, Coordination

Integration with External Data Sources

Much of the information that is captured in a groupware application is actually created there. That is, an individual enters the information directly into a groupware document. Yet most business processes rely on data that exists in other data stores in addition to the groupware object store. A groupware infrastructure should seamlessly import, share and leverage the structured data stored in relational databases and the semi-structured data found in external data sources such as desktop tools (e.g., word processors and spreadsheets), document management systems and public information networks.

- **Relational Data.** Semi-structured data often provides the context in which structured data has meaning and relevance. Inventory levels and order processing statistics tell a richer story when accompanied by illustrative details such as descriptions of product cycles, manufacturing techniques, a changing competitive environment, and customer feedback. Clearly, the integration of structured and semi-structured information is critical to the value of a groupware application and infrastructure.

By definition, the groupware object store cannot supplant the RDBMS; the design center of an RDBMS requires certain characteristics (strong locking, transactions, commit/rollback). In contrast, the design center of the groupware object store requires support for a distributed, occasionally connected model. The two technologies complement one another.

- **Image/Video Servers.** Similarly, image repositories and video servers have certain specific requirements -- namely, very large storage, and, in the case of video, the ability to deliver data in an isochronous (constant data rate) fashion. Neither a traditional RDBMS nor a groupware object store provides the appropriate vehicle for such forms of data. Nonetheless, the data/information requirements of users and groupware applications don't recognize the technological bounds of information storage and management. To better understand this, it is important to distinguish between a logical and a physical view of information. In order to present a consistent logical view of information to users or groupware applications, the object store must be the integration point of the various data sources.

- **Desktop Productivity Tools.** Microsoft has defined Object Linking and Embedding (and, to a lesser degree, Dynamic Data Exchange) as the standard for its Windows family of operating systems. A consortium consisting of Apple Computer, IBM/Lotus Development, and Novell has made progress in defining OpenDoc, a cross-platform data integration standard based on IBM's System Object Model. The common data store must comply with these standards to ensure the broadest integration with desktop products.

- **Internet and Other Public Information Networks.** One of the richest stores of semi-structured information is found in the discussion groups and World Wide Web pages of the Internet. This information represents as much of a corporate knowledge asset as any set of internal memos and discussion databases. Therefore, a complete, modern groupware infrastructure must provide a means of leveraging that knowledge. Internet resources should appear to the groupware user as native resources, and, alternatively, users should be able to publish native groupware information directly to the public network.

Updated: 02.11.95 23:35:30

Groupware - Communication, Collaboration, Coordination

Security

Messaging systems and other semi-structured data stores have traditionally used an adequate, but relatively brute force, security system. Most information in a message store or in a personal productivity tool is inherently *personal*. That is, the messages in a user's personal post office and the documents stored on a desktop computer hard drive are perceived as the "property" of the individual user. In order to ensure that only the owner of that information has access to it, the security system only has to safeguard against unauthorized access to the data store itself. Similarly, all the information contained in a public network system such as the Internet or a commercial on-line service is available to all authorized users. All that is required is initial authorized access to the service. For each of these sets of data management systems, private passwords have long served as adequate security mechanisms to protect against unauthorized access.

Groupware, on the other hand, makes use of an underlying object store from which users can "pull" information. This object store is a shared organizational resource, as opposed to the personal resource of a message store, and it typically contains sensitive and proprietary information, as opposed to the more public information typically available on public networks. The information contained in a shared object store, therefore, requires a more sophisticated security model that not only restricts access to the system at large, but which controls more granular levels of access. This is accomplished by employing multiple layers of security mechanisms: authentication, which controls access to the system at large, access control, which establishes different categories of user access to documents and information, and document and field-level encryption, which protects specific documents and fields from unauthorized viewing.

In addition, because messaging itself will continue to serve as a means of sharing sensitive corporate information, it must be a trusted courier. The integrity of individual messages -- their content and their authorship -- must be protected. Digital signatures, which rely upon the same encryption technology used for authentication, are a fourth layer of security employed by an integrated groupware platform.

Authentication

The ability to establish the identities of users as well as servers is the cornerstone of a trusted system. The functionality of other security services rests on the reliability of the authentication service. Authentication based on a system using certificates and encryption is recognized as the state of the art: the *de facto* industry standard for access to X.500 directories is the X.509 certificate, which is based on RSA public key encryption technology, recognized as the only encryption system without an exposed point of compromise.

Encryption works as follows: A user holds a certificate (or ID file) that identifies the user by name, password, license number and a private encryption key. The private key has a counterpart "public key," which is stored in a publicly accessible directory. It is virtually impossible to mathematically derive the private key from the public key. When a user attempts to gain access to a server, the following process is followed to ensure user authentication:

- The server sends the user workstation a random number.
- The client encrypts that number with the user's private key, which is resident on the workstation.
- The result is returned to the server.
- The server decrypts the number using the user's public key, which is resident in the directory on the server.
- If the numbers match, the user is authenticated and provided access to the server.

Access Control

Some people must be allowed to see certain pieces of information or entire databases, but should be excluded from other, more sensitive items. Thus, a systems administrator should be able to assign to various groups and individuals different access levels, including access to databases, documents and fields within documents. Access to each resource must be further refined to include various actions: the ability to enter, read, write, modify and delete objects.

Access control should also be flexible enough to accommodate the different "modes" that a user might assume. For example, when using a client workstation that is connected to the network, the user might have manager access to a database, but when connecting to a database from a telephone client, when authentication is not possible, the user may be granted only reader access.

Field-level and Document-level Encryption

At times a user may need to share field-level information in a document with another user while ensuring that no other users can view it. Access control can restrict field-level access to categories of users (readers, managers, etc.), but not to individuals. Therefore, for information that should be read only by specific individuals, the database designer can encrypt the sensitive information using the public key of the target readers to that sensitive field. In this way, only users with the corresponding private keys will be able to read the encrypted field. This encryption can also be used between servers so that only authorized servers can read particular documents or fields.

Digital Signatures

Users frequently have to verify that the information they receive actually was sent to them by the sender listed on the document. They also must be sure that none of the information in the document was tampered with. Verification is managed by using digital signatures. This service is the digital equivalent of a trusted courier with a wax seal. When User A digitally "signs" a message, an encrypted mathematical algorithm, or "digest" of the message is created and appended to the message using User A's private key. User B receives the encrypted document, and decrypts the digest using the sender's public key (which is available on a public directory). Using data in the message, the digest is recalculated. If the two digests match, then the sender's identity is verified. In this way, User B can be sure that the document was indeed sent by User A, and that no one has intercepted the document en route.

Updated: 02.11.95 23:38:06

Groupware - Communication, Collaboration, Coordination

Directories

One of the central components of a groupware system is the directory*,* an information store which maintains information about users and resources. As a part of the groupware object store, the directory inherits the same system-level services as the object store itself, which differentiates it from relatively simple message store directories in the following three ways: their content includes data beyond name and address, they can store information about non-human resources, they can be replicated across the distributed network.

To ensure integration and interoperability with other directories, the groupware directory should be consistent with the X.500 architecture. The groupware directory should be a rich source of information extending its role beyond the "white pages" and "yellow pages" of employee names and locations.

Specifically, they should include support for the following:

- **Rich Text.** As part of the object store itself, directories and their contents enjoy the same rich text support as other documents. Therefore, a directory document can contain such objects as user images and voice, embedded objects, and tables.

- **User Defined Fields.** The information contained within a directory should be definable by the systems administrator or end user. By defining individual fields, users can make use of a personal or private view of data that is not necessarily available to other users of the directory.

- **Linked Documents.** Because the directory itself is a document in the object store, it is possible to create links from within the directory that point to other documents in the object store. This allows administrators to "attach" important descriptive or explanatory information to an entry in the directory without incurring additional storage overhead. Linked documents help resolve the need for separate directories maintained by multiple "owners" of directory resources.

The directory should contain information regarding, not only users, but other corporate resources as well. The directory serves as an information source to users and the system itself.

- **Systems.** For some applications, the destination for a message is not a person but rather a server, a fax machine, a telephone or other electronic "endpoint." The addresses of these resources can be maintained in a directory.

- **Distribution Lists.** Electronic mailing lists can be held in the directory, to be expanded for distribution by e-mail or fax.

- **Public Key Certificates.** The directory is a suitable repository for storing public key certificates. Secured applications based on public key technology require a repository to store public keys for verifying digital signatures and for encrypting message contents among communities of users.

- **Roles.** Directories are able to associate individuals with organizational roles that can be used in workflow processes, and so workflow-based applications can utilize roles rather than specific individuals. This enables workflow processes to be more easily managed when people are on vacation or when they change job responsibilities.

- **Routing and Replication Lists*.*** The route that a message takes across hubs and routers is often determined by availability and expense, which changes depending on time of day, urgency of the message and other criteria. An entry in a directory can include descriptive information regarding routing logic, so that a particular address is always accompanied by routing instructions to ensure timely and efficient transport. Likewise, when replication is initiated (normally at some periodic interval, or at an administrator's request), the replication task looks up the location of its counterparts, and determines the most efficient way to establish a connection (via LAN connection, telephone dial-up, etc.).

The groupware directory also inherits the replication functionality of the overall system. Replication makes directory synchronization easier to implement. First, the replication process is bi-directional. Any changes made to the directory on any servers are automatically synchronized during replication, which accounts for changes, deletions and additions on both sides of the connection, as opposed to directory propagation, which only sends changes in one direction, overwriting any changes that may have already been made on the "recipient" directory. Second, directories are often large databases, and as such require significant network resources to replicate across an enterprise. The replication process should recognize which fields within the directory have been changed, added or deleted, and replicate only those changes. Replication of the entire database (or folder) would needlessly burden system resources.

Updated: 02.11.95 22:46:15

Groupware - Communication, Collaboration, Coordination

About Lotus

Lotus Development Corporation was founded in 1982 and is a wholly owned subsidiary of the IBM Corporation. Lotus offers high quality software products and support services that reflect the company's unique understanding of the new ways in which individuals and businesses must work together to achieve success. Lotus' innovative approach is evident in a new class of applications that allows information to be accessed and communicated in ways never before possible, both within and beyond organizational boundaries.

Lotus provides a comprehensive offering of award-winning products for the Windows, OS/2, DOS, Macintosh, NT and UNIX environments that are easy to use and easy to use together.

A Word on Lotus Notes

Lotus Notes is the industry leading integrated messaging and groupware product. Ironically, there is little agreement among industry analysts, customers, business partners and competitors, regarding what Notes really is. It has alternatively been described as a client/server platform for developing and deploying groupware applications, a platform for highly specialized applications, and an open environment on which independent software developers can integrate their own products.

Notes is a product that has many faces. Most people think of Notes as the applications they see and run in the Notes environment -- mail, discussion databases, etc. The power of Notes is that these applications are merely specific instances of a broad range of applications that can be written in Notes. There is no real "Notes conferencing system;" rather, there are many, many instances of conferencing systems that can be written in Notes -- because Notes is, at its core, an applications development environment. Oftentimes, someone will take an existing conferencing application and add fields to the forms so that structure can be added for a specific purpose (e.g., when you add information about a customer inquiry, fill in fields that state the details of the inquiry, the level of urgency, etc.). Once this is done, and done with ease, we have a new conferencing application tailored to a specific type of collaboration. These applications all take advantage of Notes' underlying platform-level services, such as messaging, object store, replication, security, and applications development. The Notes technology serves as a flexible groupware platform or framework upon which groupware applications can be built and deployed.

From its inception, Lotus Notes has combined three powerful group support technologies -- messaging, distributed object store, and a rich applications development environment -- to form a component architecture that serves as the basis for a large variety of groupware applications. It has been unique among groupware systems in putting forward a database model for groupware -- both in terms of the underlying user metaphor and in terms of the design tools available to end-users and IT organizations. It supports end-user tailoring tools to make simple changes to data structures and database views. It supports a rich set of APIs for developers who wish to build alternative UIs that also leverage the database model.

The implementation of the distributed database technology is also a unique advance -- still far ahead of any other groupware product. Lotus Notes' robust replication over occasionally connected networks made deployment possible at a time when few organizations had achieved full network connectivity across their enterprise. It also anticipated the growing need for true disconnected use on workstations and notebook computers.

An entire industry of independent software vendors, systems integrators, application developers and consultants has evolved around Notes. Lotus continues to cultivate this community to help ensure that customers have the resources available to install, develop, deploy and manage Notes-based applications.

Updated: 02.11.95 23:48:33