IBM MQSeries Commercial Messaging

Paul Williams
IBM UK Laboratories Limited
Hursley Park Winchester
Hants United Kingdom SO21 2JN
E-mail: kernow@vnet.ibm.com

IBM Messaging Business Solutions

The Changing Business Environment

Business organisations are re-engineering, restructuring and resizing to become more competitive, or even, to survive. Businesses are transforming themselves into flat, networked organisations and exploring new relationships with customers, suppliers and channels to deliver products and services, a concept which is described as the 'virtual company'. A contracted dynamic work force is also beginning to emerge, one that is tailored to the needs of the business.

Faced with these business and relationship changes, the underlying technology infrastructure must change. It must move away from clerical based computing, which is centralised and static, to one that is process and workflow driven, and one that is responsive to the needs of the work force in terms of providing ready access to business information that reinforces each user's expertise. Inflexible and unresponsive systems cannot sustain businesses in today's highly competitive environment. Distributed processing as described by the client/server model is a start, but disparate platforms, networks, and software are significant inhibitors to its effective implementation. The requirement is for a more sympathetic, more tolerant, more flexible technology that facilitates the development of distributed business applications.

Delivering a technology infrastructure which tolerates disparate business networks, is responsive to business change, and supports business driven events is a challenge. Increasingly, business are looking at messaging based technologies to meet this challenge. However, businesses today are entering a new era where information exchange between workgroups, business partners and the global enterprise is critical to a business's success. This places new and exacting demands on the underlying communication systems.

Vendors such as Microsoft and Lotus when describing their enterprise-wide messaging strategy are in reality talking about how they intend to use their electronic mail systems, when what is needed *today*, to assure future business operations, is a reliable industry strength message delivery system, a concept called *Commercial Messaging*, which, in addition to mail messaging, supports a range of business messaging styles which include:

- Workflow computing,
- Object messaging,
- Mobile messaging,
- Transactional messaging, and
- Data replication services.

IBM has announced and delivered a family of messaging products, the MQSeries, with the intention of fulfilling the above criteria, including mail messaging. However, *why these messaging styles*?

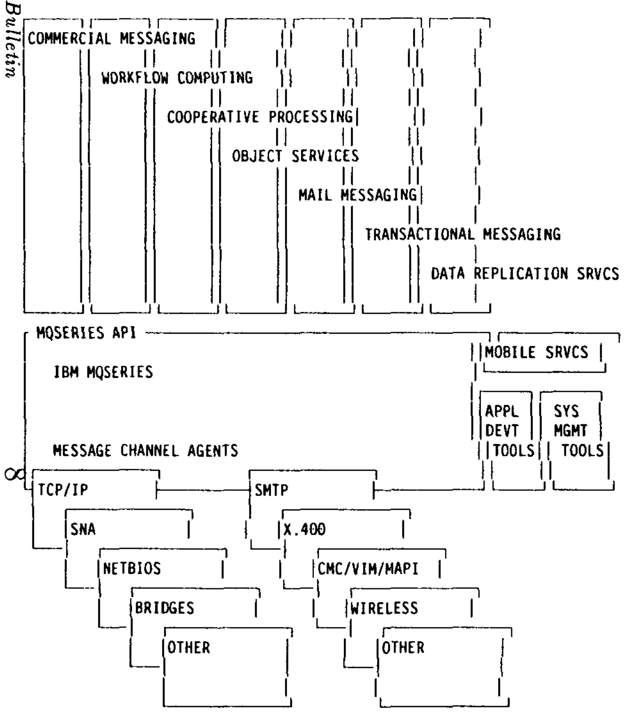
Workflow computing, object messaging, mobile messaging are the technologies of business change. Whatever the changing business organisation, there remains a requirement for business integrity and consistancy. Transactional messaging combined with data replication services addresses the need.

The business messaging styles are positioned and described in terms of the business requirements satisfied, the *IBM Messaging Business Solutions Framework*.

IBM Messaging Business Solutions Framework

The framework is intended to show the scope of IBM MQSeries enabled business solutions. Some of the solutions are available now, or will become available later in 1995 and 1996. Additional solutions will emerge as time progresses. The availability date determines the content and substance of how much can be said about each business solution today. However, the framework remains valid in terms of indicating the IBM MQSeries Business Solutions investment areas.

IBM Messaging Business Solutions Framework



The framework embraces:

- A broad range of MQSeries based business solutions
- A broad range of communications and network protocols
- Tools to help develop MQSeries based applications
- Tools to provide MQSeries systems management.

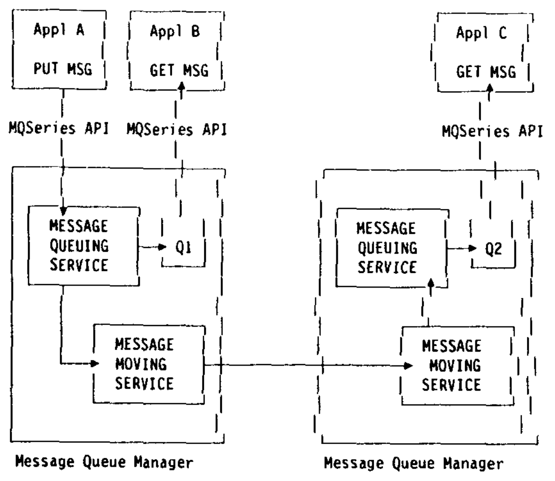
IBM Messaging Business Solutions

IBM is focusing on a broad range of MQSeries based business solutions:

- Commercial Messaging: describes the reliable industry strength message delivery attributes of IBM MQSeries, *fundamental to all the business solutions described here.*
- Workflow Computing: IBM MQSeries as the enabler of business event driven solutions. IBM MQSeries is supported by FlowMark, IBM's strategic workflow management product. IBM MQSeries provides an infrastructure which encourages the design and development of reusable software components, allowing business applications to be easily changed or modified as the business environment changes.
- Cooperative Processing: IBM MQSeries as the means to link together desktop GUI processing, server business logic and centralised host processing.
- Object Messaging: IBM MQSeries as the means to support asynchronous communications between distributed objects.
- Mobile: IBM MQSeries enhancements to provide the communications infrastructure for mobiles and wireless.
- Mail Messaging, IBM WorkGroup and IBM Mail: IBM MQSeries support for IBM provided enterprise-wide mail services; support for industry standard API's such as VIM (Vendor Independent Messaging), MAPI (Microsoft Messaging API) and CMC (Common Mail Call), to allow other applications to use IBM mail messaging services; and support to allow IBM mail messaging to connect and interoperate with other mail systems which may reside in the business organisation. The latter may be through native protocols such as SMTP, or through gateways, for example, a gateway to X.400 compliant mail systems.
- Transactional Messaging: IBM MQSeries as a full participant in IBM and non-IBM transactional environments. This provides the ability, for example, to coordinate message flows with updates to other resource managers, typically, database management systems. IBM MQSeries supports a sync point model that is particulary appropriated to loosely connected business organisations, both intra-enterprise and inter-enterprise.
- Messaging Data Replication Services: IBM MQSeries exploitation by other IBM products to provide asynchronous data replication between IBM and non-IBM relational database management systems.
- Message Channel Agents (MCA's): MCA's isolate the MQSeries kernel and API from having to understand the syntax and semantics of any communications and network protocols
- Bridges: IBM MQSeries enhancements to allow existing applications to participate without change in messaging, thus protecting their asset value.
- Application Development and Systems Management tools: IBM and non-IBM provision of IBM MQSeries application development and systems management tools.

Businesses need flexible, adaptable information technology to meet the needs of complex and fast changing marketplaces. IBM MQSeries meets that need and is playing a key role in the development of total business solutions. But first, the reliable industry strength delivery system, fundamental to all the business solutions, and characterised as *Commercial Messaging*.

Commercial Messaging



- MQSeries API shields developers from network complexities
- MQSeries API is non-blocking making it easy to perform tasks in parallel, overlap processing
- MQSeries is a connectionless technology allowing time-independent processing
- MQSeries supports assured and once only message delivery and message recoverability

What is Commercial Messaging?

MQSeries provides an industry strength, secure and reliable delivery system, a 'bet the business' delivery system. It takes responsibility on behalf of business applications for delivering messages betweem communicating partners.

MQSeries supports an indirect style of communication, the transfer of a message between communicating applications is via a *queue*. The queue simply represents a target to get to the application which is the 'other side' of the queue, the application which retrieves the message and provides the requested business function. The queue locations are transparent to the business applications. Through queue location transparency, MQSeries provides target application location transparency. MQSeries assumes the responsiblity for determining where a particular queue is located. All an application needs to know is the name of the queue which is associated with a particular service. In the above figure, it is transparent to A whether the message is retrieved by B or C. If communications are involved, MQSeries takes responsibilty for understanding the syntax and semantics of the particular protocol (TCP/IP, SNA, Netbios, ...), and further, MQSeries takes responsibility for recovering from communication failures. A business application developer is *shielded* from network complexities and can concentrate on delivering the business function.

Messages are delivered *once, and once only*. Some product implementations can generate duplicate messages. This can occur after recovery and restart from a communications failure, the products cannot determine if a message transfer ocurred before or after a failure: to play safe a duplicate message is sent and flagged as a duplicate message. This causes the receiving application to discard the message if it was presented before. MQSeries has *in doubt resolution* logic: duplicate messages are not generated.

Besides providing *assured* message delivery, MQSeries also provides message *recoverability*. The requirement for assured delivery is straight forward to understand. Many business organisations require a 'safe' delivery network, for example, a bank transferring funds from one account to another. A debit requires a matching credit. Assured delivery is essential, but more importantly if the underlying messaging infrastructure should fail, the funds transfer message must persist across restart and recovery of the infrastructure, the message must be recoverable. The provision of message recoverability further simplifies the task of the business application developer.

Assured and once only message delivery and message recoverability, are the attributes of MQSeries which make it a robust delivery system, the attributes of *Commercial Messaging*.

As mentioned above, the transfer of messages between communicating applications is indirect via a queue. From an application point of view, MQSeries is a connectionless communications technology. This means that a client application does not need to concern itself as to whether a target application, the server, is available. If it is not available, MQSeries takes responsibility for the message and delivers it when the application becomes available. As an option, MQSeries can start a target application if it is not running. This is referred to as *triggering*. This facility of MQSeries is described as *time-independent processing*.

A connectionless technology is very appropriate in a business organisation which is widely geographically distributed, or has links with other independent business organisations, or has a devolved business style. The latter is a paricular characteristic of a business organisation which has, or is, re-engineering, restructuring and resizing.

The MQSeries API is non-blocking. Control is returned to directly to the application after a message is put to a target queue. This means that the application is free to continue with other work, which may be to update a data base, or to put another message to another queue. The benefit of a non-blocking API is that multiple tasks can be scheduled in parallel from a simple application design structure, again simplifying the business application developer's task.

The MQSeries technology is implementred across a wide range of IBM and non-IBM platforms. MQSeries hides from the business applications the multivendor, multiprotocol complexity of today's business networks. MQSeries provides a *business oriented* information technology infrastructure: MQSeries based applications can more closely model the business problems that they are designed to solve making them easier to develop and maintain, the subject of *Workflow Computing*, the next subject.

Business Needs

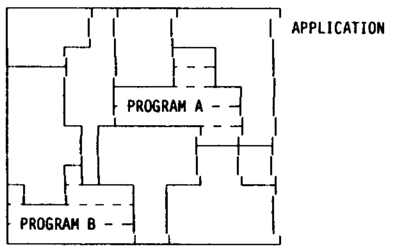
To become more competitive or even to be able to survive, businesses are transforming themselves into flat, networked organisations. Many businesses are reaching beyond their enterprise: new relationships with customers, suppliers and channels to deliver products are being explored, and contracted work forces are beginning to emerge in specialist business areas.

The IT infrastructure within these transforming businesses must change. The organisations must move away from clerical based information computing and start to provide real-time access to business data pertinent to the needs and skills of the work force.

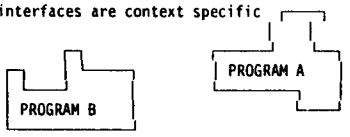
Providing this infrastructure across disparate hardware and software, and providing one which is responsive to business change is a challenge: IBM's MQSeries provides an answer.

IBM MQSeries through *Message Driven Processing* alters the way complex business applications are developed. It provides an infrastructure which encourages the design and development of an application as a set of discrete and reusable processing components. The individual components can be assembled, or extended to solve complex business problems: IBM MQSeries provides a readily adaptable framework.

What is Message Driven Processing?



Program interfaces are context specific

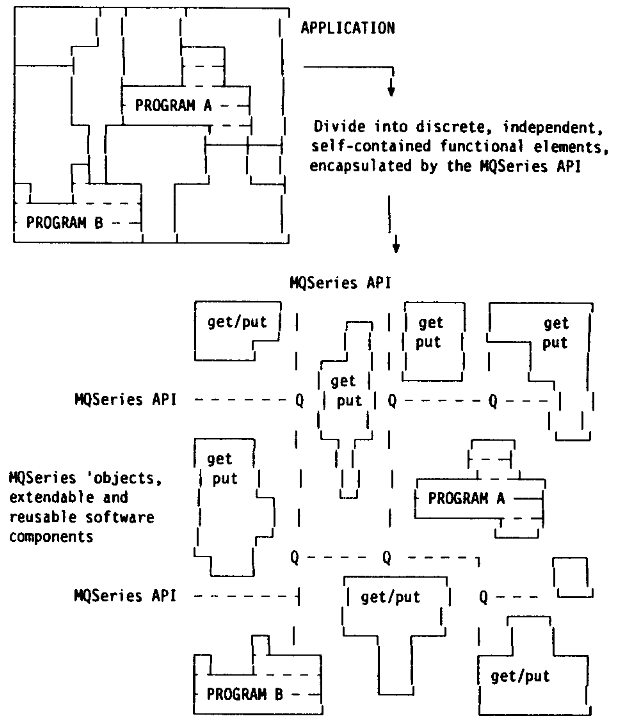


Application is business logic specific

Many business applications today are monolithic in style, written as a single large program encompassing many, if not all, of the functional elements which comprise the business application. This approach produces code which is not readily adaptive to business change: it is difficult to reuse software elements in new applications as they are encumbered by current application context; it is difficult to exploit distributed processing as the application inherits a server based/remote presentation design.

Message Driven Processing is ...

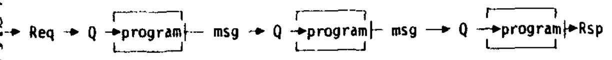
Message Driven Processing describes a style of application programming which divides a business application into discrete functional elements. The elements are the basic building blocks, software components which are considered useful and have the potential to be reusable. The components are well-defined in terms of function, input parameters, and output parameters. Input and output parameters are message flows, the software component is encapsulated by the MQSeries API.


APPLICATION
PROGRAM A
PROGRAM B
Divide into discrete, independent, self-contained functional elements, encapsulated by the MQSeries API
MQSeries API
get/put
get put
MQSeries API
Q
MQSeries 'objects, extendable and reusable software components
PROGRAM A
MQSeries API
get/put
PROGRAM B
get/put

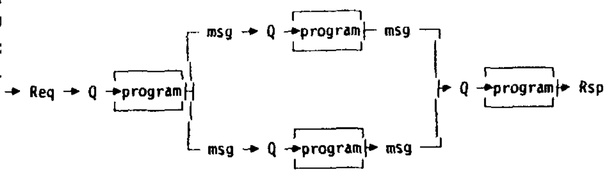
The business application is designed as a sequence of queued software components. Message Driven Processing provides a manageable and containable approach to the development of a complex business application, as each component, although contributing to the whole, can be managed as a self-contained and independent unit. A business change may cause the deletion of an existing component, or the creation of a new component, together with some change in the execution sequence of the components.

MQSeries provides the 'assembly' structure, the framework for Message Driven Processing. The 'assembly' structure, the framework, can be simple or complex, and can support a wide and rich set of business work flows:

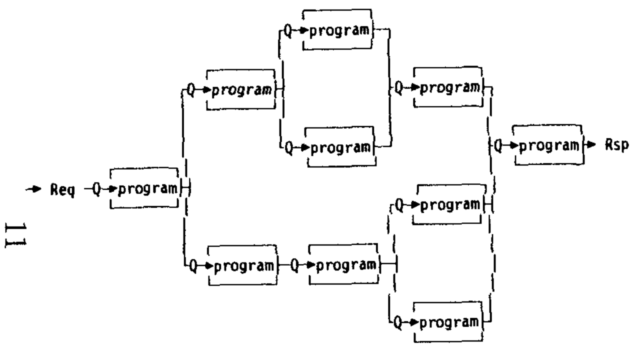
- Message chain



- Message lattice - simple



- Message lattice - compound

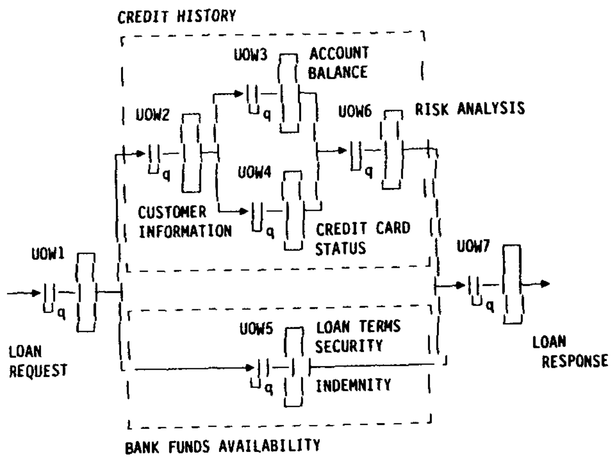


Connectionless communication and a *non-blocking application programming interface* are the attributes of MQSeries that allow the application developer to support a rich and wide set of business work flows. The connection oriented communication protocols (Conversational and RPC) and a blocking application programming interface (RPC) do not lend themselves to easily supporting work flows with splits and joins, that is, message lattices. The application programming burden may be prohibitive.

A business process which exhibits compound message lattice workflows is a bank loan:

Bank loan

The bank loan is structured as a Message Driven Processing application.



The bank loan from a customer's perspective is a simple business request. However, from the bank's point of view it is not simple but a complex processing exercise which can take many days to execute. It is a traditional clerical based business application.

The loan request causes *multiple and compound transactions to be generated* across multiple processors. There are two major pieces of work, (1), establishing the credit worthiness of the customer - account histories, credit or charge card liabilities, home loan advances, ... , and, (2), determining the business terms and conditions by which the loan may be granted - repayment period, interest rate, loan security, ... The applications which provide this information are generally implemented across multiple processors (historically, a bank has expanded its business operations into new areas by implementing specific application machines). The *multiple and compound transactions are manually generated and responses manually correlated*, a time consuming operation.

The example shows the bank loan application restructured along Message Driven Processing lines. The application is broken into discrete and independently managed software components and assembled into a MQSeries framework. Any of the individual components can be changed or modified without affecting the rest of the processes in the framework. For example, modification of the risk analysis module dependent on customer status, high risk for student loan, low risk for valued customer. In the bank loan example, Message Driven Processing improves the bank's productivity by automating what was previously a clerical or procedural based activity, and improves customer service by allowing the banker to concentrate on the business rather than the procedures to execute the business. The business benefits of Message Driven Processing are evident. However, it raises some questions:

- What, or who, determines the execution sequence, the need for decomposition and recomposition?

If the 'rules' are contained in the program then it defeats the objectives of Message Driven Processing which is to make business changes easier by avoiding expensive application code changes, and to produce code which is inherently reusable in new business applications.

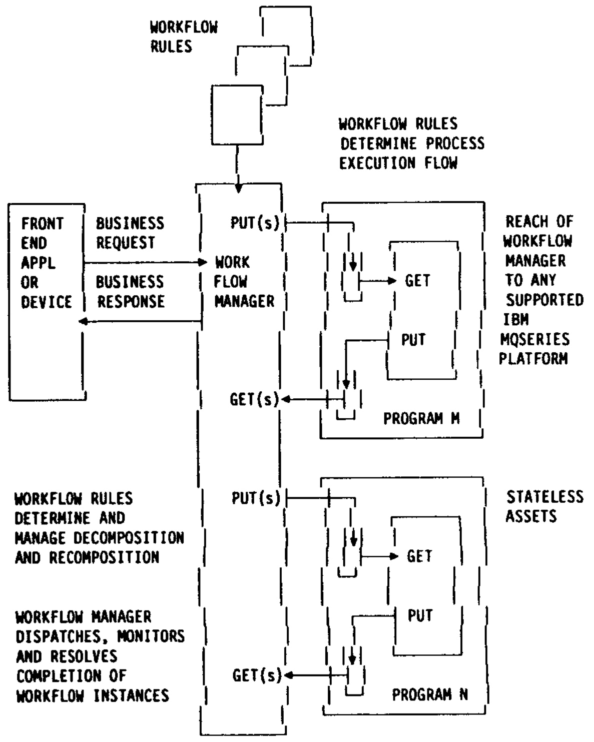
- What happens if one or more of the programs are unavailable, or fail and cannot be recovered?

  Well, it depends on which programs fail, critical or not. Does the whole business application have to be aborted if one or two of the programs fail, or can a partial completion of the application be tolerated? What, or who decides?

The need is to make the individual software components stateless, that is, make them independent of the particular business application instance in which they are cooperating.

The question is: *how does Message Driven Processing work?*

How does Message Driven Processing work?


WORKFLOW RULES
WORKFLOW RULES DETERMINE PROCESS EXECUTION FLOW
FRONT END APPL OR DEVICE
BUSINESS REQUEST
BUSINESS RESPONSE
WORK FLOW MANAGER
PUT(s)
GET
PUT
GET(s)
PROGRAM M
REACH OF WORKFLOW MANAGER TO ANY SUPPORTED IBM MQSERIES PLATFORM
WORKFLOW RULES DETERMINE AND MANAGE DECOMPOSITION AND RECOMPOSITION
PUT(s)
GET
PUT
GET(s)
PROGRAM N
STATELESS ASSETS
WORKFLOW MANAGER DISPATCHES, MONITORS AND RESOLVES COMPLETION OF WORKFLOW INSTANCES

The execution sequence, and its management and control, is defined by a set of *workflow rules*. The workflow rules are interpreted and executed by a *workflow manager*. The workflow manager dispatches, monitors and resolves completion of workflow instances, it provides the state management.

The workflow rules are a set of predefined instructions for controlling the execution flow for the processing of a business request. The rules may determine that multiple processes may need to be invoked to satisfy the business request, *decomposition*, from which, multiple multiple replies may need to be assembled into the business response, *recomposition*. The workflow manager is the first point of capture of a business request. It identifies the business request and executes the appropriate workflow rules, it provides the management and control of the business processes. MQSeries provides the infrastructure to manage concurrent and overlapped processing, is tolerant of process unavailability, and through assured and once only message delivery and message recoverability provides robust message delivery services.

Messsage Driven Processing business benefits

Businesses need flexible, adaptable information technology to meet the needs of complex and fast changing marketplaces. Expensive, rigid legacy systems cannot sustain the demands of profit driven business organisations and their shareholders. Such a rigid structure also places an impossible burden on an information systems organisation's ability to respond and deliver new powerful business applications quickly.

MQSeries and Message Driven Processing encourages the design and development of reusable software components. Message Driven Processing provides a manageable and containable approach to the development of a complex business application, as each individual software component, although contributing to the whole, can be managed as a self-contained and independent unit. In response to business change it is simpler to delete or create components, or change workflow rules, than to amend large monolithic code structures. MQSeries and Message Driven Processing delivers a a technology infrastructure which is responsive to business change, and provides the base for business event driven solutions.

Relationship with business process modelling

Many business organisations have developed business process models which at the enterprise level define entity relationships which correspond to the major business functions. At lower levels the logical entity relationships may be defined for particular applications or for specific modules within applications. MQSeries allows computing resources to be configured such that they more closely match the business process model, the workflow rules are a dynamic expression of the logical business entity relationships. MQSeries provides a technology infrastructure which is *in touch* with the business organisation.

Workflow Management Products

Workflow management products that support MQSeries are:

- IBM FlowMark for AIX and OS/2
- IBM Business Event Processor (BEP)

  BEP is a CICS/ESA Solutions Offering which integrates IBM MQSeries for MVS/ESA
- Early Cloud & Company Message Driven processor (MDp)

  MDp is a CICS/ESA Solutions Offering which supports IBM MQSeries for MVS/ESA as an option. Early Cloud & Company is an IBM MQSeries Business Alliance partner, corporate headquarters in Newport, Rhode Island, USA.

Cooperative Processing Product

Why is enterprise client/server difficult?

The section *Workflow Computing* introduced the concept of *Message Driven Processing*, delivering a technology infrastructure for business change, business process re-engineering and distributed processing. It also positioned workflow computing in terms of a business process model, which at the enterprise level defines entity relationships which correspond to major business functions. At lower levels the logical entity relationships may be defined for particular applications or for specific modules within applications. The business process model can provide a map to the distributed process model.

- The argument has been made that a messaging technology like MQSeries allows computing resources to be configured such that they more closely match the business process model.

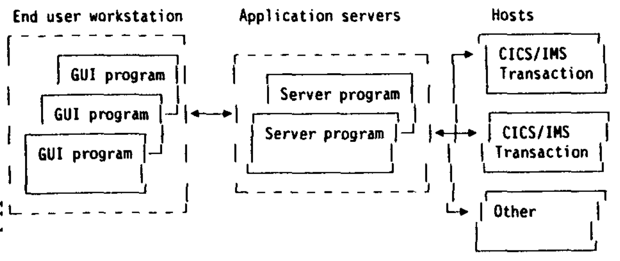
However, distributed processing as described by the client/server model is not fulfilling the expected and hoped for return on investment: new powerful business applications are not being delivered quickly, And why not? At the enterprise level, client/server is difficult.

So, what contributes to the complexity? At the enterprise level it is the sheer numbers, the product of the number of people and the number of application services to which they need access. At the departmental level, the integration of applications and data and providing access is more straight forward, the simple LAN client/server model. However, it is not scaleable to multiple server DB's, and legacy systems. The programming culture is 'serial', 'one-at-a-time', 'synchronous', 'connection oriented', 'RPC-like'. It creates comfort, a 'know what is happening' familiarity. It is adopted by standards bodies such as DCE, CORBA and AD tool vendors. It is a constraining programming culture.

Business process re-engineering creates its own problems, such as, process management, event notification, business control, concurrency, state management. These are all issues which reinforce the need to change the underlying technology infrastructure if it is to avoid business applications assuming the solution burden, which combined with a constraining programming culture is the very reason that enterprise client/server is difficult.

An MQSeries business solution is the provision of an integrated Application Development/Application Execution and Systems Management environment for advanced client/server applications.

Cooperative Processing - enterprise client/server

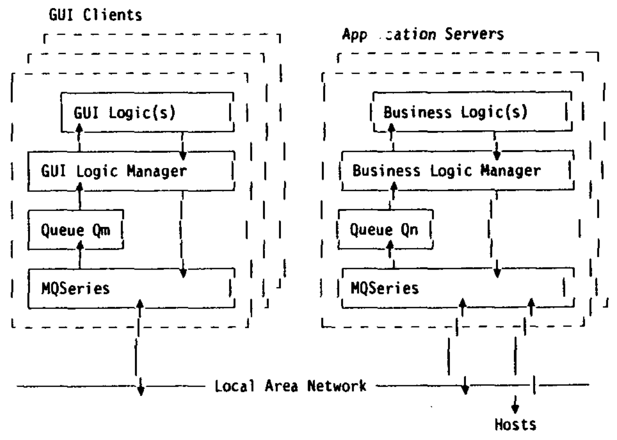


The solution enables a three tiered distributed logic model, encapsulates business rules to manage complex GUI and server processes, and understands interprocess business relationships.

Client, server, host connectivity is via MQSeries. The reasons for using MQSeries are:

- To design and develop extendable and reusable software
- To provide recovery and integrity facilities
- To coordinate the commit and recovery of multiple queue operations within a unit of work
- To provide a seamless logical network - workstation to host
- To support a rich and wide set of business workflows.

The key solution components are:



The Business Logic Manager provides an infrastructure similar to that created by OS/2 Presentation Manager and Microsoft Windows, in the sense that the system monitors for relevant inputs and invokes applications as appropriate. The Business Logic Manager provides an infrastructure suited to Message Driven Processing Applications. The Business Logic Manager continuously monitors its input queue (MQSeries queue) for messages. Rules define what message combinations and application states need to exist to invoke a business application. As rules are satisfied, the Business Logic Manager invokes the appropriate business application.

Similarly, the GUI Logic Manager continuously monitors its input queue (MQSeries queue) for messages. However, the situation is somewhat different for GUI applications as there is a need to integrate the asynchronous arrival of MQSeries messages with the event-driven structure of a GUI program. A GUI program registers its event queue with the GUI Logic Manager. (In OS/2, an event queue is a Presentation Manager message queue). Based on the business event, the arrival of the message, the GUI Logic Manager places a message on the appropriate input event queue. The GUI program is responsible for retrieving the message and calling the business application (the GUI application). The binding between the solution and a GUI development tool (such as VisualAge) makes this mechanism transparent to the application programmer who only needs to concentrate on the business application function.

The Job Viewer component runs on the client workstation and provides a container window that lists the windows associated with a job and allows navigation between them. The Job Viewer allows a user to run multiple jobs without getting confused between the windows of one job and those of another.

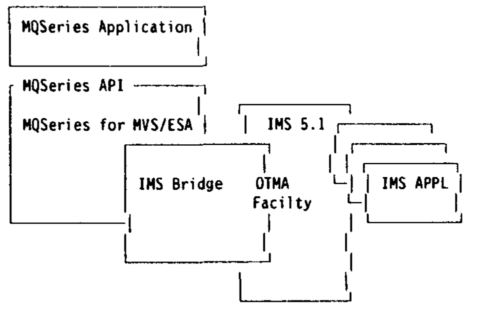
Solution characteristics summary

The solution encourages and allows the human to stay in control. The human prescribes the activities, initiating work as the business opportunities present themselves: it is 'negotiable', or 'collaborative'. The solution provides the required navigation and management services, executing GUI, server and host based processes as appropriate (typically in real time). The navigation and management services extend to message generation, tracking and correlation services, and state management of the activities invoked.

The solution provides a robust infrastructure for developing flexible structured client/server applications: It supports the design, development and execution of powerful applications for *customer facing workers*, for example, travel agents, estate agents, bank staff, insurance brokers

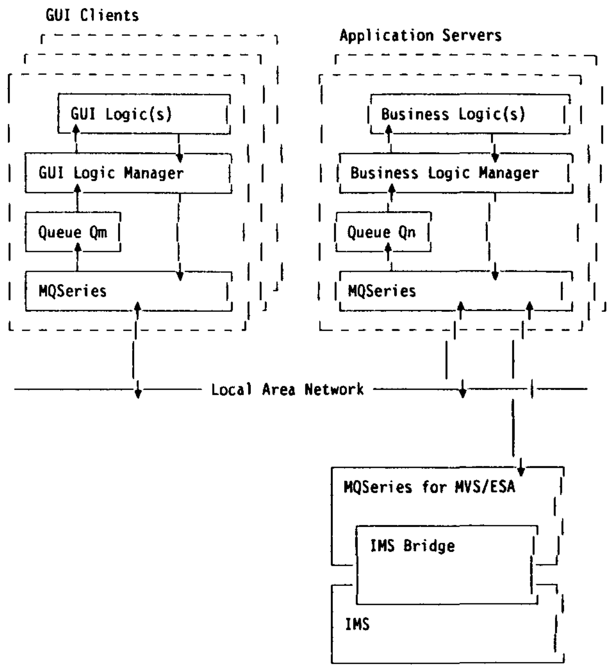
Cooperative Processing and legacy systems

Bridges in the context of MQSeries describes MQSeries enhancements which allow existing applications to participate without change in messaging, thus protecting their asset value. An example is the MQSeries bridge to IMS.



- The IMS 5.1 Open Transaction Manager Access allows MVS applications to submit IMS transactions. The IMS applications run unchanged (GU, GN, Insert) as if the transactions were entered from a real device.
- MQSeries for MVS/ESA is providing a bridge to OTMA. This allows any MQSeries application to submit IMS transactions.

The Cooperative Processing Solution in conjunction with MQSeries provided bridges addresses business process re-engineering needs at the enterprise level. An example shows the MQSeries bridge to IMS.



IBM MQSeries provides the infrastructure which facilitates enterprise client/server. In this example, legacy IMS based data is easily made available to a new GUI program. The Cooperative Processing Product and the MQSeries bridge to IMS removes infrastructure considerations: the application programmer can concentrate on the business function; the user, the employee, can concentrate on delivering business service.

Cooperative Processing Product

Product general availability (outlook):

- OS/2 server and OS/2 and Windows 3.1 clients 6/95
- AIX server 9/95 (target 6/95)
- MQSeries bridge to IMS 4Q95.

Mobile Messaging

What is Mobile Messaging?

The benefits to the enterprise of providing mobile computing to its employees are often among the easiest to understand from a business perspective: increased number of customer visits per day by the field force, reduced back office expense by decreasing the need to transcribe from paper forms to data entry systems and, sometimes, even the potential to reduce office real estate.

Most enterprises will have to re-engineer part of their business processes to take advantage of mobile computing. The underlying technical infrastructure must change as well. It must meet the requirements of a total end-to end solution and not just the problems associated with the mobile-fixed network boundary. The need to integrate and support mail, other workgroup products, and business applications in an intermittently connected environment typical of many mobile scenarios, is a natural fit with the capabilities and facilities of MQSeries.

The system used by the mobile worker will in many cases be comparatively lightweight. MQSeries coverage will be expanded to provide messaging capabilities on platforms from PDAs upwards. In very resource constrained environments this will require a reduction of the messaging facilities compared to a full function (and likely fixed) MQSeries server.

The design of mobile solutions must bear in mind that 'mobile' describes the behaviour of the people involved rather than simply the portability of computer systems. The general case is the need to support a worker who may start with a workstation connected to a LAN, move to using a wireless connection in the field and then use a telephone line, say from a hotel. MQSeries support will be enlarged to cope with all these environments. This is more than simply enlarging the range of communications protocols supported.

Over time capabilities will be provided for intelligent transmission. For example, to avoid the situation of transmitting a large multimedia document while the user is connected to a slow, comparatively unreliable wireless link, rules based transmission support will take into account criteria such as: length of message, estimate of cost, priority, quality of service of the link. These rules can be modified by the business user.

The MQSeries mobile support will enable business organisations to design profiles for single users, or groups of similar users, to establish and vary transmission rules, cost management and other features, for example, security access, by the location of the user. 'Locations' can be defined by the customer, for example, in the office, hotel, field. The user can alter the location to reflect their current working environment. A qualitative reflection of cost management will enable the enterprise to set limits, for example, on cellular radio usage by field operatives, that can be viewed on the mobile unit. When the limit is hit, transmission of messages can be suspended or an emergency credit can be used.

One of the great strengths of MQSeries is that the capabilities of mobile clients will be compliant with MQSeries servers in the fixed network. However, to reflect the resource constrained nature of the systems, some special enhancements may also be provided. For example, to avoid intalling large directories on the mobile device, name resolution of queues may be deferred until the message is transferred to a server on the network. As most mobile system users will not normally need all the possible MQSeries facilities at the same time, installation options will be provided to install subsets.

Object Messaging

What is Object Messaging?

Many business organisations are moving towards Object Technology to reduce the time it takes to develop applications, and to design applications which are responsive to business change (to be competitive in the market). Some of these organisations have distributed business requirements that are satisfied by MQSeries. It is planned to provide access to MQSeries facilities from object oriented languages.

MQSeries directions (1)

- The MQSeries Object Model

  The model will provide an object oriented programming interface to the procedural MQSeries API (languages being addressed include C++ and SmallTalk). This allows users to integrate messaging into their products without any constraints on object-oriented functionality. (The MQSeries Object Model is also part of IBM's Mail Messaging and Mobile Messaging solutions).

IBM Object Technology

IBM is playing a leading role in the development of an Object Technology environment that will change the economics and simplify the process of software development for business organisations.

The System Object Model (SOM) is the cornerstone for IBM's offerings in the emerging multiplatform, distributed object computing environment, and is available today in IBM's SOMobjects Toolkit. SOM defines an infrastructure for sharing objects. It allows objects to be packaged in a way that exposes only their interface. As a result, an object can be written in one language and used or refined by another, so for example, a component written in C++ is usable and easily refined by Smalltalk. SOM addresses the pervasive need for cross language support in object development.

SOM scales up gracefully to support objects that are distributed across a network. The interface for a distributed object is identical to a local SOM object. The SOMobject Toolkit contains Distributed SOM (DSOM) (an Object Request Broker (ORB) that complies with the CORBA standard set by OMG). DSOM services locate the remote object and route requests and responses, masking the complexity from the developers and end users. DSOM supports important industry network transports which include TCP/IP, IPX and NetBios, synchronous communications protocols: it is planned to enhance DSOM with MQSeries support.

MQSeries directions (2)

- DSOM over MQSeries Transport

  Some business organisations have identified the need for an object oriented open systems solution which can cope with heterogeneous platforms, asynchronous calls and disconnected message delivery. DSOM over MQSeries transport meets this need. It will also allow DSOM to exploit MQSeries industrial strengths: assured and once only message delivery and message recoverability.

- DSOM to MQSeries Bridge

  Interoperabilty is required between DSOM and existing MQSeries applications to enable DSOM programs to take advantage of such applications without having to change those applications or their operating environment. Such interoperability gives DSOM programmers access to existing MQSeries applications on any MQSeries supported platform.

- OMG Event Services

  OMG Event Services provides a supplier/consumer model. Suppliers and cosumers register with 'event channel'. It is planned to provide an event channel variant which again exploits the industrial strengths of IBM MQSeries.

Transactional Messaging

What is Transactional Messaging?

The participation of messaging in a transaction processing environment is described as *transactional messaging*. The MQSeries technology is implemented as a resource manager, akin to a data base manager. Whereas a data base manager manages changes to data base records, MQSeries manages changes to queues. In a transaction environment, message flows can participate in unit of work management. Changes to queues, message additions and deletions, can be coordinated with changes to other resource managers. MQSeries can participate in the coordinated commit and recovery of transactions which may include, for instance, changes to an SQL data base, and changes to queues. A simple example:

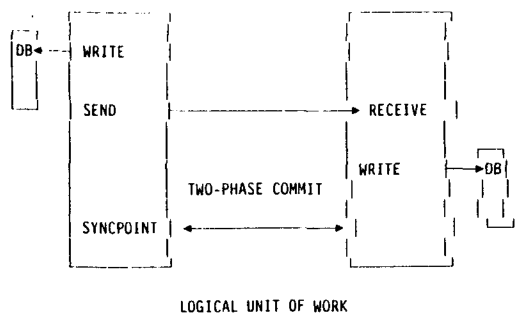
- A transaction (CICS, IMS, Tuxedo, Encina, ...) is written which updates an SQL data base (DB2, Sybase, Oracle, ...) and informs an interested party that the update has occured by putting a message to a target queue.
- It is not desirable to flow the message, for example, if the SQL update fails: erroneous information is disseminated.
- If the SQL update and the message put is coordinated within a unit of work, then if either action fails, the other action is rolled back. In the example, if the SQL update fails, the message does not flow to the target queue.

Another example may be that it is necessary to inform a number of targets of an event, multiple message flows. The requirement is that all recipients must learn of the event, or none must know. This is a typical requirement in a trading floor where, perhaps, all the traders must learn of a market event if it is to avoid one or more traders taking advantage of a business opportunity at the expense of others, for example, an arbitrage opportunity. Multiple puts of a message to multiple target queues can be managed within a unit of work. If the application, for example, which is generating the multiple puts fails before the target queue list is exhausted, any puts to a partial list are rolled back, no messages flow, a 'level playing field' can be maintained.

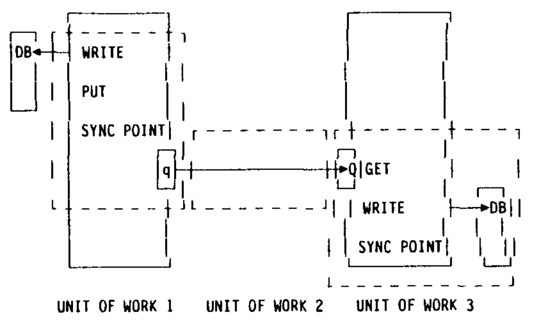
Transactional messaging provides an alternative commit coordination model to that provided by the distributed two-phase commit model, exemplified by implementations of IBM's Distributed Relational Base Architecture (DRDA) , and somtimes described as the 'conversation model', where changes to multiple and distributed resource managers are coordinated within a single logical unit of work. Many businesses do not require, or they may not permit, or it may not be technically possible to coordinate changes to multiple and distributed resourcese within a single logical unit of work. The requirement is for the assurance that within an acceptable window of time, the changed resources arrive at a consistant state.

The conversation and messaging models are contrasted.

Synchronous conversation model



Asynchronous messaging model



Many business organisations have a requirement to provide data integrity and consistancy as applications make changes to recoverable resources across multiple transaction and data base systems. That is, make changes to multiple resources appear as a single unit of work. If the participating systems are unlike, or if there is no guarantee that the systems are concurrently available, it may be impractical to coordinate a logical unit of work through a two-phase commit protocol. For example, even if the participating systems are similar, delays in the execution of the two-phase commit process may cause locks to be held for an unacceptable period of time with a resulting adverse effect on performance. Messaging starts to provide an asynchronous alternative for maintaining data integrity and consistancy.

A business application, using messaging, can divide the single synchronous unit of work into multiple asynchronous units of work, changes to messaging resources, the *puts and gets*, being coordinated with changes to other resource managers, such as DB2, by the local sync point manager.

NOTE: *The logical unit of work scope does not apply to the execution of the function associated with the message, but rather to the queuing of the message.*

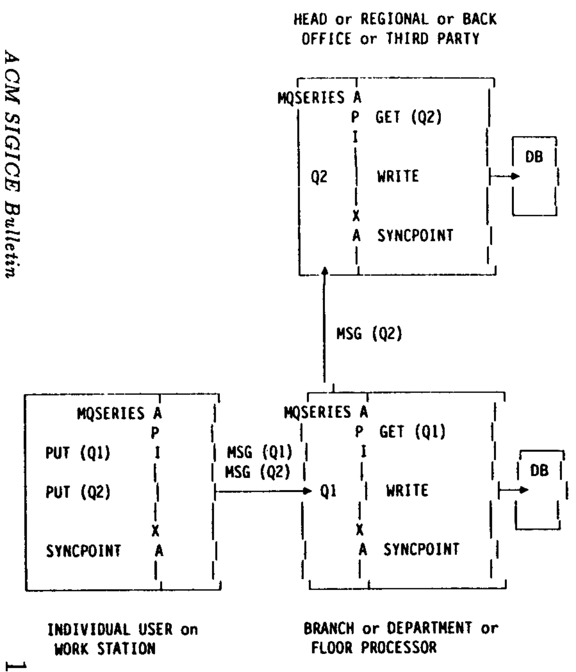
Figure description:

- Unit of Work 1 - under local syncpoint control the 'write' to the data base is committed and the message is secured on q, a local 'transmission' queue which is transparent to the 'putting' application;
- Unit of Work 2 - the message is safely delivered and secured on Q, the destination queue, the target of the 'putting' application;
- Unit of Work 3 - the message under local syncpoint control is deleted from the queue and the 'write' to the data base is committed.

NOTE: As the messaging approach may involve multiple units of work, there may arise a business context where an application is required to 'undo' changes to some resources. This is an application responsibility, as the underlying messaging infrastructure cannot have knowledge of the business context which causes some of the multiple units of work to be 'undone'. (The knowledge to determine if an 'undo' is required can be given to a workflow manager which is described later).

Many business environments deploy mult-tiered or peer coupled processors interconnected by multiple networks within the business organisation (intra-enterprise) or across business organisations (inter-enterprise).

Multi-tiered or peer-coupled networks



- Multi-tiered or peer-coupled networks
  - Trading floor and back office
  - Plant floor operations and production planning
  - Hospital patient management and insurance claims
  - Travel agent - airline, hotel and car rental coordination
  - Railway freight management and control

Common business data is maintained in multiple processing environments. As applications make changes to this data, there is a requirement to provide data integrity and consistency, but not within a single logical unit of work. The requirement is to assure that at some acceptable time the changed data is brought to a point of consistency compatible with the business expectations.

For example, a customer of a travel agent expects airline, hotel, and car rental reservations to be consistant with his or her travel itinerary, and be assured that the reservations are made. There is no business requirement to coordinate the reservations (the data changes) within a logical unit of work.

Another example is in the trading industry. Trade capture data is required for position management on the trade floor, and by the trade confirmation, settlement, clearing operations in the back office. Both the trade floor and the back office operations need to reflect a consistant state of the business, however, there is no requirement for immediate consistency as the business operations are asynchronous. The trading floor operation is dynamic, driven by the market. A trader needs to understand his position second by second. The back office is accountable to various regulatory bodies (internal and external) whose reporting requirements do not share the immediacy of the market.

Transactional messaging combined with assured and once only message delivery and message recoverability, simplify the application design. Considerable programming burden is removed from the application:

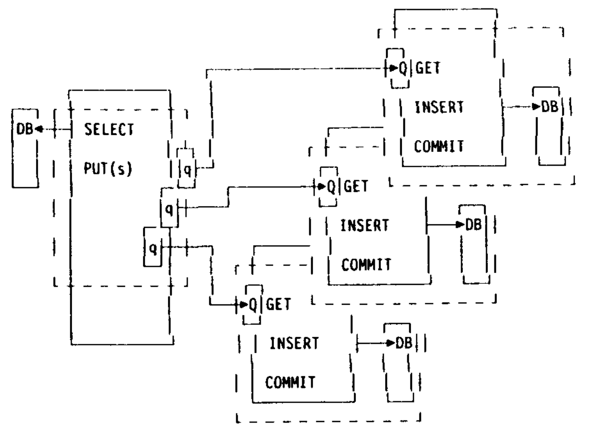
- It does not take responsibility for message delivery
- It does not take responsibility for protecting messages from a major platform or communications failure
- It does not take responsibility for protecting messages from local application failures, nor coordinating the commit and recovery of business transactions which include both communications and data base resources.

MQSeries Commit Coordination

MQSeries resource commit coordination can be provided by:

- CICS/ESA, AIX CICS/6000, CICS OS/2
- IMS/ESA
- OS/400 Commit Control
- An X/Open XA compliant transaction monitor.

Messaging Data Replication Services



SERVER CLIENTS

What are Messaging Data Replication Services?

As businesses re-engineer, restructure and resize a common need is to have the flexibility to distribute and process data as appropriate within the organisation. A typical requirement, for instance, is to replicate, or propagate, data from a corporate relational database management system (RDBMS), quite often DB2, to clients, perhaps in a workstation environment running an RDBMS from one of a number of different suppliers (IBM, Oracle, Sybase, ...). There may be several reasons for data replication:

- Distribution of data to business partners (suppliers, distributors, customers)
- Off-load work to smaller, lower cost processors
- Improve and provide consistant user data access.

MQSeries through assured and once only message delivery and message recoverability provides a robust data delivery system and simplifies application design.

However, the question may still be raised as to how to get protection from local application failure, or coordinate the commit and recovery of message flows with SQL updates.

Transactional Messaging assumes the existence of a transaction manager to provide the unit of work management and resource commit coordination. This may not always be available, or desired. There may be no business requirement to instal a transaction manager. This raises the question as to how to provide the required management and coordination without a transaction manager. A direction is that MQSeries is enhanced to play a resource commitment coordination role.

IBM Data Management and Distribution Products

IBM data management and distribution products that exploit MQSeries include:

- IBM Data Collection and Distribution (DCD)

  DCD is a component of the IBM Enhanced Financial Platform. It supports collecting data from various, heterogeneous sources, storing the data in a centralised 'Enterprise Warehouse', and then distributing the data, as required, to various client databases and end users, the business professionals.

  Data is is often stored in various databases and file systems, accessed by different methods. The diversity of systems, tools and data access languages make data management a complex task. DCD hides this complexity from the business professional, and provides at his/her workstation, customised data to support particular business needs. DCD uses 'IBM MQSeries for MVS/ESA' to ensure the data distribution is accurate and consistant. DCD distributes data on demand, or time of day.

  DCD supports data collection from relational, non-relational databases and flat files (MVS based), and deposits the data in a common relational database (DB2). Selective data distribution is to OS/2 (OS/2 DBM and DB2/2) and MVS clients (DB2). Future environments will include AIX/6000 and UNIX and support for DB2/6000 and Oracle.

  The Sumitomo Bank, Tokyo, has adopted IBM Enhanced Financial Platform DCD to allow end users to retrieve and consolidate enterprise wide data. It has a prominent role in a 60 billion yen investment in a new Information Processing System for the Bank.

- IBM Data Replication Services Offering, Integrated Systems Solutions Corporation (ISSC), Boulder, Colorado

  The service offering provides for the planning, design, installation and configuration of IBM Data Replication products that include: DataPropagator NonRelational, DataPropagator Relational and DataRefresher. ISSC has the skills to extend the scope of IBM Data Replication products to include data distribution between non-IBM data bases, exploiting the industry strengths of MQSeries for the data distribution. MQSeries support.

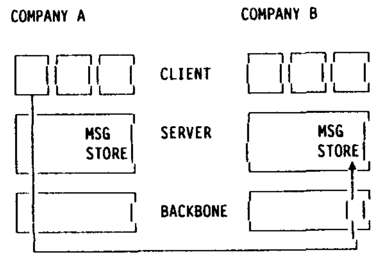
Mail Messaging

Mail and Messaging

Mail and messaging have become confusing terms, and in the views of many they are synonomous, views encouraged by vendors of mail products.

First generation mail messaging solutions packaged client, or user agent, and the mail messaging service together in a structure where the client could not be separated from the service. The client created the message and passed it off to the mail service through a proprietary application programming interface. The mail messaging service managed the distribution, typically files in a shared file system.

The next generation of mail messaging solutions have begun to embrace newer technologies like: client/server, objects, messaging service or backbone. The client/server structure is evolving where the server can function as a messaging service not only for its clients but for other vendor clients, and provides far more than simple store and transport of mail elements. Messages can be a combination of objects (text, voice, video, program, commands, ...) that go between people, a person and an application, or between applications.



This layered messaging structure demands that messaging solutions be built from the infrastructure out, with more focus on the messaging backbone than on the client.

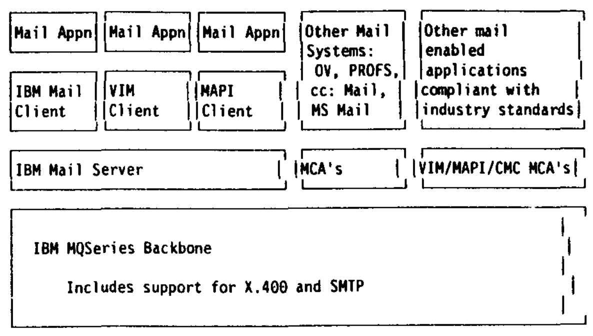
However, as businesses embrace a multitude of applications that communicate through messaging services, messaging is no longer a simple interpersonal communication application, but a *mission critical application service* on which the business operates and is dependent.

This is the dichotomy between mail and messaging. When mail vendors discuss enterprise-wide messaging strategies, they are describing how they intend to use their client focussed electronic mail systems, when what is needed today, is a reliable industry strength message delivery system.

A key element of IBM's messaging strategy is the introduction of IBM WorkGroup. IBM WorkGroup is a portfolio of activities that allows organisations to coordinate people, activities and applications. IBM WorkGroup uses MQSeries for its distribution services. One of the main applications is Mail. The IBM Mail solution, therefore, avoids the mail and messaging dichotomy and shares the benefits of Commercial Messaging demanded by a business critical application portfolio.

IBM Mail

IBM Mail based on MQSeries provides a protocol independent, open and expandable solution:


Mail Appn
Mail Appn
Mail Appn
Other Mail Systems: OV, PROFS, cc: Mail, MS Mail
Other mail enabled applications compliant with industry standards
IBM Mail Client
VIM Client
MAPI Client
IBM Mail Server
MCA's
VIM/MAPI/CMC MCA's
IBM MQSeries Backbone
Includes support for X.400 and SMTP

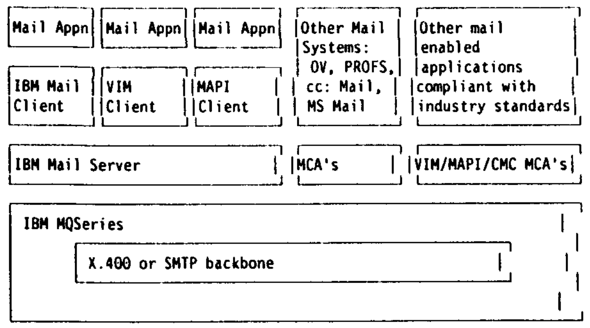
- Uses one common enterprise-wide 'industry strength' delivery system which is secure and reliable, MQSeries.
- Supports other vendor mail clients via industry standard interfaces: VIM, MAPI and CMC.
- Connects other mail enabled applications through VIM, MAPI, CMC.
- Supports other mail systems such as OV, PROFS, cc:Mail, MS Mail.
- Supports X.400 and SMTP transports.

The IBM Mail solution allows for 'plug and play' flexibility in choosing messaging components.

IBM Mail - Other Vendor Mail Systems

The IBM Mail solution will run over (tunneling) and/or connect (insertion) to other vendors' systems through native protocols such as SMTP and X.400.

Tunneling



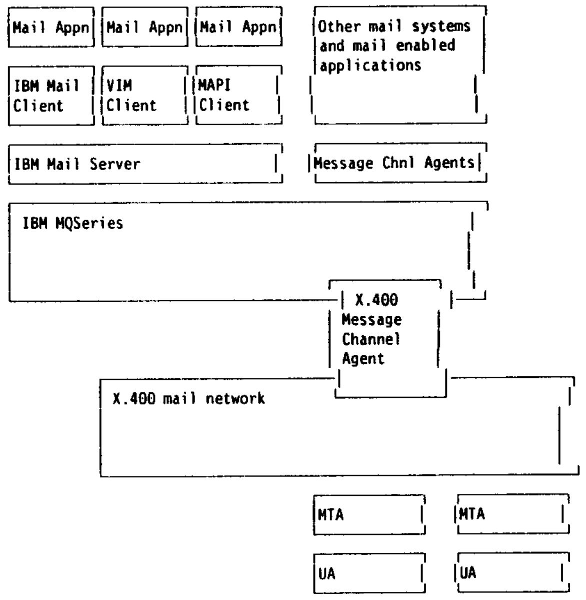
- MQSeries provided Message Channel Agents for X.400 and SMTP

  Brings MQSeries strengths to existing mail networks.

NOTE: The transfer of MQSeries messages over a mail backbone network is not confined to mail messages. Any MQseries application can exploit the MQSeries X.400 and SMTP agent support.

Insertion

The following figure shows the IBM Mail network exchanging messages with a X.400 based mail network.



Message Channel Agent unwraps MQSeries message, handles mail message conversion before submitting to the X.400 network.

- 'MS Mail' is a trademark of Microsoft Corporation
- 'cc: Mail' is a trademark of Lotus Development Corporation.

NOTE: Not part of WorkGroup, but it is appropriate to reference it here, is IBM MQSeriesLink for Lotus Notes.

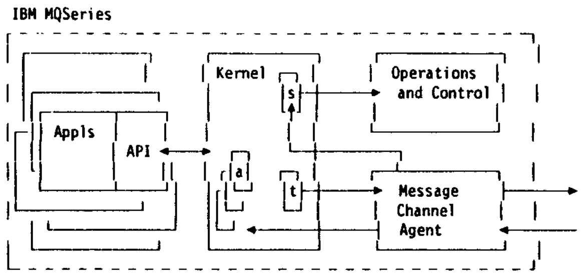
A Notes application consists of many objects, one of which is a Notes form that is created within a Notes environment. The form contains several type of fields. The fields define methods for obtaining the data that fills the fields. MQSeriesLink provides an extension to a field definition describing how the data can be obtained and stored.

MQSeriesLink extends the Notes reach to include in any Notes form, data which may reside on any MQSeries supported platform.

MQSeries - Generic Structure

The MQSeries design policy is *open and cooperative*. It provides well defined interfaces and protocols to several key functional software components which are interchangeable and replaceable. This facilitates and supports third party cooperation in delivering comprehensive MQSeries functionality.

MQSeries - Generic Structure



```
Legend: a Application Queues
        s System Administration Queue
        t Transmission Queue
```

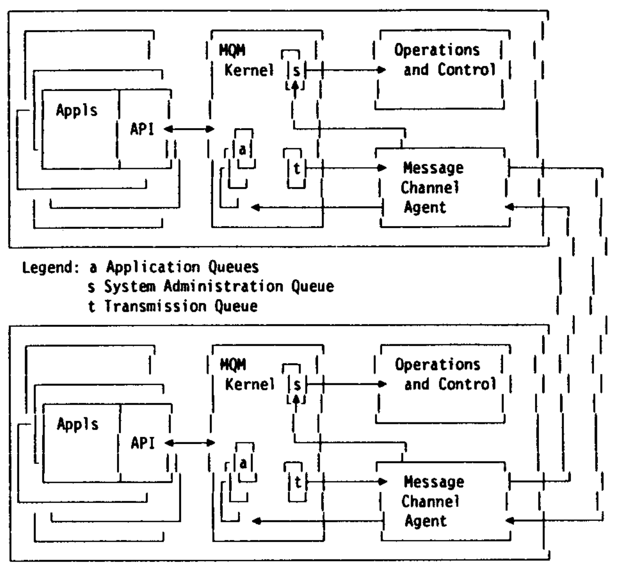
IBM MQSeries supports the MQSeries Application Programming Interface (API) and provides ther following generic function:

- Queue name-to-address resolution

  Simply, where is the location of the queue, that is, the MQSeries node in which the queue can be found. How to get there is the responsibility of message routing.

- Message routing and Message Channel Agent (MCA)

  Messages destined for another node are placed on a transmission queue which represents the connection to an adjacent node. The adjacent node may be the destination node, or an intermediate node on the route to the destination node. MQSeries identifies a particular transmission queue via a routing table (the transmission queue is transparent to the application). Multiple transmission queues may be defined between adjacent nodes. This may facilitate the segregation of message traffic, for example, batch versus interactive traffic.



```
Legend: a Application Queues
        s System Administration Queue
        t Transmission Queue
```

Only the MCA can read transmission queues. It transfers the messages to the adjacent node. The MCA also receives messages from an adjacent node. The messages are either placed on a local queue, if the destination is the node of the receiving MCA, or on a transmission queue to be forwarded to the next adjacent node on the route to the destination.

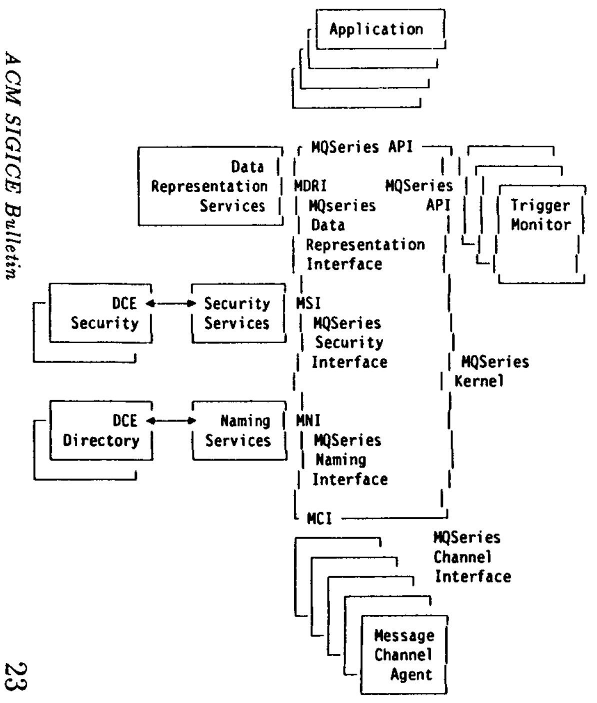
- Administration

  System administration panels enable an administrator to create, delete, monitor, display and control MQSeries resources.

- Unit of work participation

  MQSeries is a participant in unit of work management and resource commitment coordination.

MQSeries - Open Structure



The MQSeries structure is open. It provides interchangeable and replaceable functional software components which include the following:

Message Channel Agent

The MCA's isolate the MQSeries kernel and API from having to understand the syntax and semantics of any communications and network protocols. This provides MQSeries great flexibility and allows it to easily support any communications and network protocol based on business need. Alternative or additional MCA's can be provided by third parties through the MQSeries Channel Interface.

Naming Services

The service are invoked to provide queue name-to-address resolution. MQSeries provides built-in naming services, one is the optional use of DCE directory services. Alternative name resolution services can be provided by third parties through the MQSeries Naming Interface.

Security Services

MQSeries provides security services via an External Security Manager. The services are invoked to provide user identification and authentication, and access control to MQSeries resources. MQSeries provides optional support for DCE security services. Alternative security services can be provided by third parties through the MQSeries Security Interface.

Data Representation Services

MQSeries provides a basic level of data description and conversion services for user data. These can be replaced or enhanced by third parties through the MQSeries Data Representation Interface.

Trigger Monitors

As an option, when a message arrives on a destination queue and the application designed to retrieve the message is not running, MQSeries can start the application. This is referred to as triggering. Basic triggering policies are provided by MQSeries. Trigger monitor routines can be replaced by third parties to provide alternative triggering policies: the trigger monitors use the MQSeries API.

MQSeries - Application Development Tools

A number of application development tools are, and will, support the generation of MQSeries applications.

IBM VisualGen Support for MQSeries

The current version of VisualGen works with MQSeries. VisualGen includes a sample set of MQSeries applications that correspond to the C and COBOL sample set made available with MQSeries.

The sample applications include the following reusable parts that can be incorporated directly into business applications without change:

- Definitions for MQSeries API parameters, both simple items and structures
- Definitions of MQSeries API constants and return codes
- Statement groups (VisualGen subroutines) for:

  Initializing MQSeries API parameters to default values

  Calling MQSeries API entry points.

The sample applications will execute in a number of MQSeries environments (see below). It is planned to describe the sample applications in a technical report and make them available through the VisualGen forum.

VisualGen's single system development and test environment makes it a powerful productivity tool for building MQSeries applications. The developer can define and checkout the logic of all the applications (in the test environment) with a single and local MQSeries node before the applications are generated for the production MQSeries environments.

Environment Support: It is planned to support VisualGen generated MQSeries applications in the following environments in 1994:

- MVS/ESA environments, CICS, IMS, TSO and batch
- CICS VSE
- CICS OS/2
- OS/2
- OS/400
- AIX/6000.

Additional MQSeries environments will be supported.

IBM VisualAge support for MQSeries

Support for MQSeries is planned.

Network Software Associates product support for MQSeries

Network Software Associates (NetSoft/NSA), Arlington, Virginia, USA, are providing software products that enhance several application development tools with MQSeries support.

- VB Connection for MQSeries

  VB Connection for MQSeries, an NSA product, enhances Visual Basic to allow programmers to develop Windows applications that communicate with host and server applications using MQSeries.

- PowerBuilder and other environments

  NSA is producing a Windows Dynamic Link Library (DLL) that provides a call interface to the MQSeries API. The DLL allows any development environment, or macro language, that supports DLL calling conventions to send and receive data using MQSeries. This includes PowerBuilder, Word, Excel, Lotus 1-2-3 for Windows, and others.

  – Visual Basic, Windows, Word and Excel are trademarks of Microsoft Corporation

  – PowerBuilder is a trademark of PowerSoft

  – Lotus 1-2-3 is a trademark of Lotus Development Corporation.

MQSeries - Systems Management Tools

MQSeries Systems Management Requirements

MQSeries Systems Management is the process by which the availability of MQSeries services to applications is controlled. It is separate and distinct from the application environment, since it is concerned with the operation of MQSeries in a reliable and predictable fashion rather than with the specifics of the business services provided. The mechanism by which this control is exercised is the management system.

The provision of MQSeries Systems Management covers a range of categories which include:

- MQSeries Installation and Configuration

  The ability to instal multiple MQSeries products (on multiple operating platforms) and define the MQSeries resources and attributes (queue manager. queues, processes).

- MQSeries Operational Control

  The ability to operate the MQSeries environment, that is, maintain system throughput and availability, provide first level of system problem actions and rectification, and operate from a centralised control point. Tasks include:

  - Starting and stoping MQSeries queue managers
  - Starting and stopping traces
  - Managing the log and log control data sets
  - Managing backup and recovery.

- MQSeries Administrative Management

  The ability to manipulate (alter, define, delete) MQSeries objects from a centralised control point.

- MQSeries Problem Determination

  The ability to perform problem determination tasks from a centralised control point:

  - Collect records of detected error events from the MQSeries nodes
  - Collect and interpret statistical data about permanent and temporary errors
  - Generate alerts to the appropriate notification point about potential problem situations
  - Recommend possible user activities to relieve detected problems

- MQSeries Performance Management

  The ability to monitor the performance of the MQSeries environment, that is, diagnose, modify, and tune.

- MQSeries Change Distribution

  The process by which changes are applied as opposed to Change Management which is the process of requesting, planning, recording changes. Change Distribution is concerned with, for example, installing new versions or releases of MQ, applying defect maintenance to the MQSeries environment.

- Administrative Management, of which, the major elements considered are:

  - Problem Management
  - Change Management
  - Configuration Management.

  Administrative Management (as distinct from MQSeries Administrative Management) is the process by which information about problems, changes, and component configuration is managed and documented

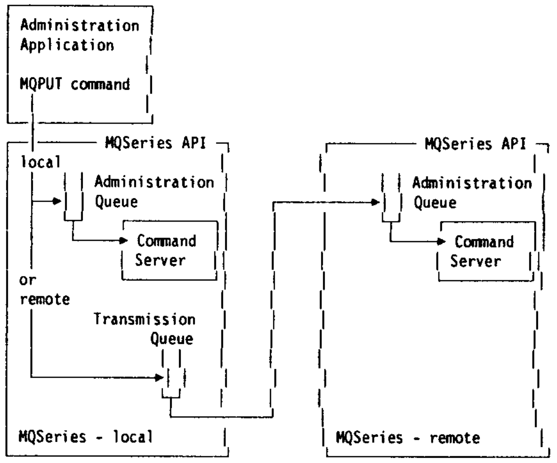
Each of these categories represents a set of functions needed to effectively manage a MQSeries network, regardless of the operationg platform environments, interconnecting network protocols, and their physical locations.

Some of these functions are provided as part of the MQSeries family, others by industry accepted systems management applications.

MQSeries Systems Management Provisions

MQSeries provides local and remote support for basic configuration, operational and administration services.

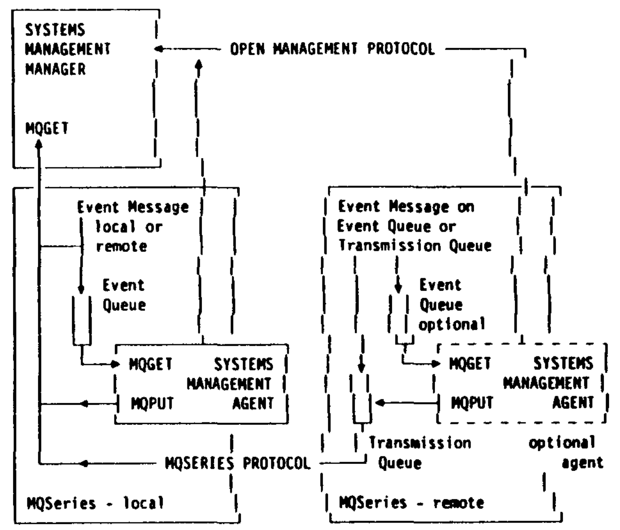
Each MQSeries is configured with a command input queue, known as the systems administration queue, and a MQSeries provided command server which processes commands (messages) on the administration queue. The commands are generated by an administration application which is supplied with MQSeries. It can also be supplied by a third party.



Note: Remote support is the ability to manage an MQSeries node from any other (chosen for the purpose) MQSeries node, that is, the provision of a single point of control. This support is not currently available across the whole MQSeries family: it is a planned direction.

Enhancements are introduced in the areas of problem determination and performance management services, sometimes referred to as *instrumentation and alerts*. Instrumentation and alerts is the facility to monitor various MQSeries thresholds and error conditions, and when detected, generate a special event message, giving details of the conditions detected, and place it on a MQSeries event queue. (There are several categories of event queues).

Fundamental to the MQSeries systems management approach is the concept of *managers* and *agents*. The manager in this context is the systems management application that manages the MQSeries resources, getting agents to act on its behalf as a result of some event. (An example of a systems management application is IBM's NetView). Systems management agents are MQSeries applications programmed to the MQSeries API. The agents retrieve messages from the event queue(s) and forward to the systems management manager.


SYSTEMS MANAGEMENT MANAGER
MQGET
OPEN MANAGEMENT PROTOCOL
Event Message local or remote
Event Queue
MQGET SYSTEMS MANAGEMENT AGENT
MQPUT
MQSERIES PROTOCOL
MQSeries - local
Event Message on Event Queue or Transmission Queue
Event Queue optional
MQGET SYSTEMS MANAGEMENT AGENT
MQPUT
Transmission Queue
optional agent
MQSeries - remote

The systems management agents may communicate with systems management managers using an open system management protocol, or MQSeries.

There have been several initiatives in establishing systems management standards:

- Simple Network Management Protocol (SNMP)
- OSI Common Management Information (CMI)
- X/Open and OSF Distributed Management Environment
- IBM SystemView.

It is anticipated that managers and agents will be IBM and third party provided supporting one or more of the standards.

Summary

The MQSeries approach, to providing MQSeries systems management facilities, reinforces the MQSeries open and cooperative policy of providing well defined interfaces and protocols to facilitate and support third party participation in delivering comprehensive MQSeries functionality.