An Inclusive and Extensible Architecture for Electronic Brokerage

Jenny Hands
Fretwell-Downing Data Systems Ltd.
jhands@fdgroup.co.uk

Mikhail Bessonov, Ahmed Patel
University College Dublin
{mikeb, apatel}@net-cs.ucd.ie

Ron Smith
KYROS
rsmith@compulink.gr

Abstract

The output and experience of a large European project in Electronic Brokerage called "Generic Architecture for Information Availability" (GAIA) is presented. The paper describes a reference model and functional architecture for value-added mediation in Electronic Commerce. The customers are provided with a uniform way of accessing heterogeneous suppliers without changes in the supplier software. A way of employing distributed objects for large scale service integration and multi-enterprise transactions is shown. The issues encountered during the implementation of the CORBA-based pilot prototype and the application of the architecture in 3 diverse domains are presented.

1. Introduction

Dictionaries define commerce as "the buying and selling of goods and services".

Electronic Commerce can be broadly seen as applying the information technology and telecommunications advances of recent years towards increasing the efficiency, effectiveness, and functionality of traditional commerce practices in the supply chain linking producers of good and services and the eventual consumer. As such, Electronic Commerce covers a diverse range of commerce related activities including but not limited to:

- Processes that aid the potential customer (individual or business), in locating the goods and service they need and at the same time allowing suppliers to make potential customers aware of their products.
- Processes facilitating the negotiation of a basis for the transaction that mutually benefits all parties
- The processing of the agreed transaction
- The management of the on-line delivery of the good or services from the supplier to the consumer
- After sales services that ensure that the customer is satisfied with the order and delivery process and is able to use the delivered goods or services to their full extent.

The objective of this paper is to examine the role of brokerage within the context of Electronic Commerce and to present the Generic Architecture for Information Availability (GAIA[1]), an expandable architecture for the implementation of electronic brokerage systems.

1.1 Electronic commerce concerns

Electronic Commerce offers a foundation for many benefits for both the consumer and the supplier. The fact that DELL Computers profitably sells over $5M a day via their WEB site is proof of the benefits that Electronic Commerce has to offer customers and suppliers [8]. In parallel there are also risks associated with the introduction and spread of Electronic Commerce. Outside of the well-publicized issues, such as those in regard to credit card fraud, there is a real danger in what we will refer to as supplier overload. With the ease at which the current state of the art allows the creation of a WEB site, one can easily envision the situation where every individual or organization producing goods and services sets up its own on-line WEB store. If such an unstructured uncontrollable approach to the spread of electronic commerce continues, the future success and up-take of the technology and concepts is questionable. Related business models, architectures, and standards must be developed and adopted if Electronic Commerce is to fulfill its promise.

1.2 The need for the electronic broker

With this avalanche of on-line suppliers and information about goods and services, how is the consumer to locate, purchase, and obtain the goods and services they seek, at a fair market price, and from a supplier they can feel confident with? The introduction of the role of an on-line electronic broker into the electronic commerce supply chain linking consumers and suppliers presents the ideal solution to this problem. Mediation by these on-line electronic brokers joins customers and suppliers thus stimulating the market

[1] This work is part funded through the European Commission's ACTS Programme.

activities of both parties. Electronic brokerage systems have the potential of offering suppliers the capability to expand the number and type of potential customers for their products thus allowing small and medium size suppliers to more effectively and efficiently compete with larger suppliers in their domain.

1.3 The need for a supplier independent architecture for electronic brokerage

In order for the electronic brokerage concepts to be effectively implemented it is key that such implementations be based on relevant architectures and agreed standards. The brokerage architecture must be scaleable and be applicable to the distributed on-line nature of today's electronic commerce supply chains. It must handle the diverse nature of the existing and future goods and services to be traded. At the same time it must handle the heterogeneity of the systems deployed by the actors involved in the supply chain and the heterogeneous networks linking them. The architecture must allow implementations that are commercially feasible. It must support the ability of actors taking the role of broker to add value to the supply chain and to financially benefit from this added value. The architecture must support the transaction oriented ACID properties which are currently absent in today's most commonly used Internet technologies [16]. To ensure success, the architecture must be generic and applicable to a diverse range of domains. And most importantly, to ensure fair competition and protect the consumer from a monopolistic environment the architecture must be supplier independent.

The explosion in Internet access and the advent of electronic commerce presents great potential for the market for electronic brokerage services. To date, most efforts towards the realization and standardization of this market have focused on infrastructural issues such as secure on-line payment, electronic product catalogues, and digital delivery. However, if there is to be interoperability between businesses in the supply chain, permitting participation of both large and small suppliers and promoting choice and diversity, a supplier-independent architecture for brokerage meeting the above described requirements is required. The Generic Architecture for Information Availability (GAIA), presented in this paper, is being developed by an international consortium of nineteen organizations. The GAIA approach supports an information society in which Information Brokers offer service economies and refinements that enhance the supply chain to the advantage of producers and service providers in diverse domains.

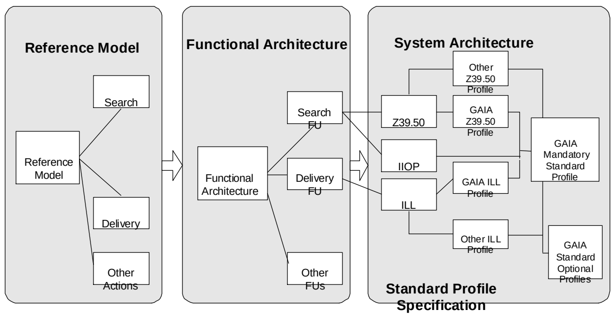

Figure 1. GAIA Standard Formulation Levels of Abstraction

2. The GAIA project

2.1 Conception of the GAIA project

The GAIA project was conceived in 1996 to address the requirement of the on-line business society for a standards-based brokerage architecture, as outlined above. Around the central theme of the project, to produce an architecture for brokered information supply services, objectives were defined to establish a GAIA Standard with architecture and protocol recommendations, and a generic toolset to contribute to the development of practical, interoperable information brokers. In this way, GAIA would facilitate the location and purchase of online information, goods and services.

The GAIA consortium comprises service providers from diverse fields in IT, telecommunications and commerce, and a number of research organisations.

2.2 Overview of GAIA's work

In the remainder of this paper, we present the key results of the GAIA project with reference to their application and applicability in electronic commerce.

As shown in the following Figure 1, the GAIA environment has been modeled from a number of perspectives, differing in their level of abstraction.

At the most abstract level, the GAIA Reference Model provides a common basis for the description and specification of brokerage systems. The GAIA Reference Model is defined in terms of the Roles, Actions, Events and Entities involved in electronic brokerage. The GAIA Functional Architecture defines the functional elements of the GAIA system, embodying the GAIA Reference Model. The Functional Architecture specifies the roles and relationships between the GAIA Services which instantiate the Actions and supporting Events of the Reference Model. Based on the functional relationships of architectural elements and their relationship to the world at the large, the GAIA Standard is specified.

The development of generic tools within the architecture has been a core activity in the project, providing key reusable services, including user access components and mechanisms to use data from existing repositories. The GAIA pilot implementation broker is built on the CORBA distributed object architecture. The experience the GAIA project gained from building a CORBA-based Broker is described later in this paper.

Crucially important to the development and validation of the GAIA approach has been the involvement of representative bodies and user communities in trials of the GAIA service. GAIA selected three target domains in which to demonstrate various implementations of the developed GAIA compliant brokerage system: the Music industry, the Technical Components industry, and the Publishing industry. The Music Industry Trial focuses on delivering information and product in specialized genres (namely dance, ambient, and free improvisation) to the sector's highly dispersed global consumer base. The Technical Components Industry Trial provides electronic brokerage services to allow engineers to select and retrieve technical information on a wide selection of die for the design of MCMs (MultiChip Module) and hybrid circuits. The Publishing Industry Trial allows students, researchers, and industry experts to locate, order, and receive digitally a wide range of on-line material, including scientific articles from journals and agricultural information such as agricultural statistics and customer contacts.

The span of these domains, from the structured, highly specified, standards oriented, technical nature of the Technical Components industry to the artistic, loosely structured, multimedia nature of the Music industry, is to ensure that the GAIA architecture covers the needs of a wide array of other domains.

2.3 The GAIA Reference Model

Before describing the Functional Architecture, which is a key result of GAIA and the basis for deployment of brokerage services, we will briefly outline the Reference Model, which is the highest level of abstraction considered by GAIA, in order to clarify the use of terminology.

The core concepts promoted by the GAIA project's reference model are the roles of customer, broker and supplier, and the actions of search, locate, order and deliver. These generic roles and actions have been found to be a good basis to discuss and develop a working demonstrator broker infrastructure, with brokers and suppliers spread around Europe. The scope of the reference model has been limited to these areas of electronic brokerage and has not been formulated to address other areas of brokerage such as workflow and pre and post sales support. Clearly the brokerage system must interact with back-of-house systems supporting such functionality.

Central to the reference model is the GAIA broker, which acts as a locator and supplier of information or services to customers, i.e. users who seek services and information online, and likewise a distribution mechanism for suppliers wishing to promote goods and services. The broker either supplies the information or service to the customer itself (in which case it is acting in the supplier role), or else (playing a customer role itself) it sources the information or service from another broker/supplier. Information about online services or information (generically called products) is maintained by brokers, so that they know where to locate the products required by the customer. The information maintained by a broker may be arbitrarily specialised, with special brokers for particular domains or geographical regions. Brokers may be federated, thereby extending the domain or geographical coverage of individual brokers.

The function of the broker is to provide a path whereby a customer may find and obtain a product. If the broker knows where such a product can be found, he supplies the customer with the location information. If the broker does not know the location of a product, he requests a search on his behalf from other brokers, thus widening the search. In order to allow a search to be carried out, the terms of the search need to be clearly defined. A description needs to be given by the customer to the broker of the product that he requires. The use of metadata is central to the functioning of the GAIA broker, both for the customer to describe the product that he requires, and for the GAIA broker to propagate the search to further brokers.

Whilst brokerage involves various phases of business operation [3], including awareness creation etc., it is the transaction phase that is of most concern to GAIA. The "actions" which the GAIA architecture caters for during a transaction are

- Search, where the customer describes the nature of the product, and the broker returns the identity of products matching this description
- Locate, where the customer obtains from the broker the location and terms of supply for a required item
- Order, where the customer requests supply of the required item from the given location
- Delivery, where the broker causes the item to be sent to the customer

Around these main actions are supporting actions, of which the most important are authentication, tariffing, and connection to payment systems.

Based on these concepts, a more detailed architecture can be developed, first in terms of functional decomposition, and then in terms of systems architecture.

3. GAIA Functional Architecture

The GAIA functional architecture decomposes the overall functionality of the brokerage system into a number of components, and describes the roles and relationships of the components, and the manner in which they interoperate. The key elements of the GAIA functional architecture are the *GAIA kernel*, *Functional Unit Managers (FUM)*, *Functional Units (FU),* and *abstract primitives (AP)*. These are described in detail below.

3.1 Functional Units

The brokerage system provides a number of services to its users. These services are supported by the functions of the brokerage system. These include, for example,

- searching
- metadata collection
- format translation

Each of these functions can be provided by a number of different candidate technologies. However, the operations that are required to be carried out remain the same - regardless of the selected technologies, the functional requirements do not change. The required operations are described in terms of abstract primitives, which can be mapped to the protocol requests of whatever technology is selected to support the function. A mapping component, called a Functional Unit (FU), is defined for each candidate technology, and converts calls to abstract primitives into protocol instructions. The FU acts as an interface between its particular technology and the rest of the brokerage system.

Functional Units are defined for each candidate technology that can be used to fulfil a particular functional need of the brokerage system. A Functional Unit accepts abstract primitive invocations, and maps them to calls to the particular technology to which it is dedicated. The results of the calls to the technology are translated into the corresponding abstract primitives and returned by the FU.

3.2 Functional Unit Managers

As noted above, a number of different candidate technologies can be used to fulfil a particular functional requirement of the brokerage system. Depending on the details of the GAIA transaction (underlying network, Customer system capabilities, domain requirements, etc.), different technologies may be more useful during different Transactions. As a result, each candidate technology has its own Functional Unit, which is invoked when that particular technology is required.

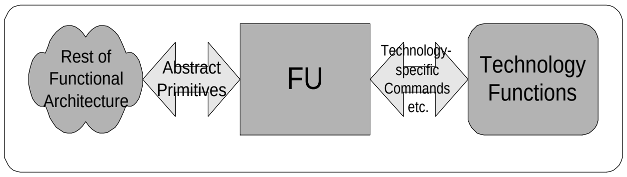
Rest of Functional Architecture
Abstract Primitives
FU
Technology-specific Commands etc.
Technology Functions

Figure 2. GAIA Functional Unit

A number of different Functional Units can exist which fulfil the same functional requirement of the brokerage system. In order to select the most appropriate FU (and technology), the brokerage system needs to know which is most useful at any particular time. This is the responsibility of the Functional Unit manager, or FUM. Each function of the brokerage system has a single FUM, which is invoked in terms of abstract primitives by the Broker kernel. This FUM selects the most appropriate of the candidate technologies, and calls the corresponding FU. The interface between the FU and the FUM is defined in an open, platform-independent, programming-language-independent manner. It is important to notice that all the FUs subordinate to the same FUM have **the same** interface, expressed in terms abstract primitives, and also specified in CORBA IDL (IDL is used to define the interfaces in an unambiguously fashion). This holds true even in the case of considerable differences between the technologies used to implement these FUs. This hides the internals of particular FUs from the FUM. For example, a new FU implementing some recently emerged search protocol, can be developed by a third party vendor and incorporated into the broker without any modification of any other component. This makes the architecture easily extensible.

3.3 The Kernel and Abstract Primitives.

The kernel of the brokerage system acts as a buffer between the FUMs, and as a bus for the transmission of abstract primitives between FUMs. It also acts as a repository of local information, and as a shared data store for FUMs. All calls to abstract primitives are executed via the kernel, which exports all the abstract primitives imported by the various FUMs, and imports all those exported by the FUMs.

Communication between the GAIA kernel and FUMs, and between FUMs and Functional Units, is carried out in terms of abstract primitives. In a pure CORBA world we would call them just methods of particular published interfaces. However, to allow a non-ORB based implementation of a GAIA broker and to keep the

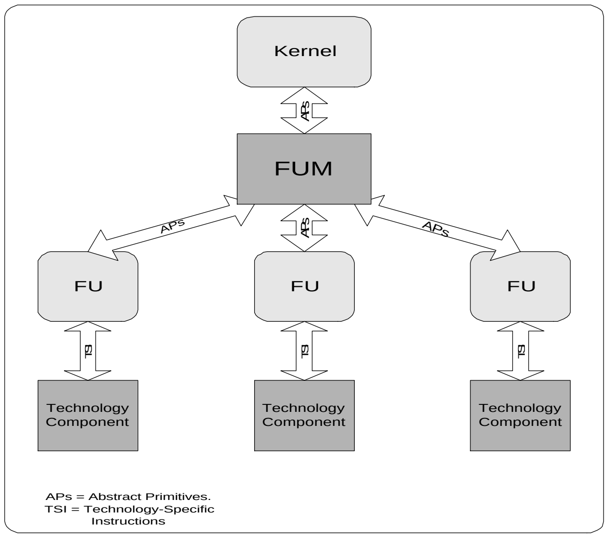

Figure 3. Communication between FUM and FU

interface general, a concept of abstract primitives was introduced. These abstract primitives correspond to a particular operation which a FUM will carry out on behalf of the kernel, or which the FUM expects the kernel to carry out. Each FUM imports a set of abstract primitives, representing those services which the FUM expects to receive from some other part of the system. The services which the FUM is prepared to provide to other elements of the brokerage system are presented in the form of exported abstract primitives. All abstract primitives are imported from, and exported to, the kernel. The kernel acts as a bus for abstract primitives. Abstract primitives are also used in communication between the FUM and its FUs. The FU exports abstract primitives to the FUM.

3.4 Elements of the GAIA Functional Architecture: Description of FUMs

The core activities of the brokerage system include:

- searching for and identifying information Products that fit a user description
- sourcing information Products the identification of which is known
- allowing users to order Products
- delivering information in file format
- delivering information as a continuous media stream
- monitoring use of the system and charging for it
- enforcing the security policy of its particular security domain
- collection of information about what is available (metadata collection)
- providing a user interface to the brokerage services and alerting users as to the availability of information, using email, mobile terminals, etc.
- interacting with external directory services.
- allowing the user to provide feedback on a particular Action or transaction.

This is illustrated in terms of FUMs in Figure 4. The Table 1 contains a brief description of every FUM specified by the GAIA Functional Architecture.

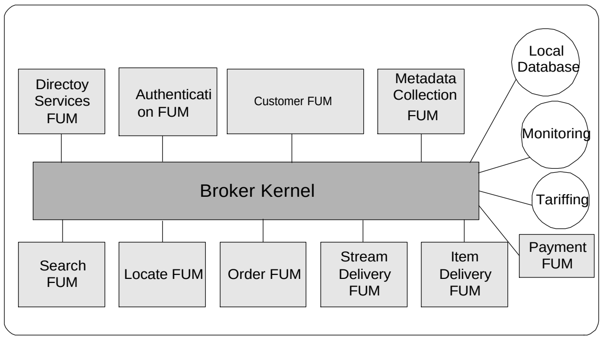

Figure 4. GAIA Functional Architecture FUMs

In the event that a GAIA broker is a distributed entity, with different FUs and FUMs running on different platforms to the kernel, inter-module communication uses the CORBA model. The internal operations of the broker (the abstract primitives imported and exported by the FUMs) are encoded using the IIOP [6] inter-orb protocol, along with any data or other variables required for the operation, and then transported among the FUMs. The pilot implementation developed by the GAIA project supports this distributed model.

4. Choice of Brokerage protocols

4.1 The GAIA Standard

As previously discussed, the architecture of the GAIA brokerage system is designed so that a number of different technologies can be used to fulfil any particular functional requirement of the system and furthermore, the GAIA Broker can interoperate with non-GAIA systems where compatible technologies have been implemented. However, in order to promote interoperability, GAIA also specifies a “GAIA Standard”, in which one technology is selected for each FUM.

The criteria for selection reflected the over-riding need to specify a workable standard, with the best chance of producing brokerage systems that are interoperable with other information navigation systems. Key selection criteria include the following:

- popularity and acceptance
- standards status
- functionality

Table 1. FUM Responsibilities Summary

FUM	Responsibilities
Search	Accepts requests to carry out a search for Products that fit a particular user description. It returns lists of identifiers of Products that fit the description.
Locate	Accepts Product identifiers, and discovers where they may be obtained. It returns lists of Suppliers and locations for the Product.
Order	Manages negotiations between a Customer and a Supplier, in order that agreement may be reached on the terms of availability of a particular Product or group of Products. Following the negotiation phase, the order FUM accepts purchase commitments from the Customer and forwards them to the Supplier. It returns a notification of the status of the order Action.
Item Delivery	Manages the delivery of file-structured items to the Customer.
Stream Delivery	Manages the delivery of real-time multi-media data streams to and from the Customer.
Payment	Provides a mechanism for payment from one actor to another.
Authentica tion	Provides a mechanism which allow a user to prove his identity to the brokerage system.
Metadata collection	Supports the collection of Product descriptions and where they are available
Customer	Provides an interface for the user to allow him to interact with the brokerage system. It also alerts him when a Customer-specified event occurs.
Directory Services	Provides an interface between an external directory service and the brokerage system.

- feasibility of integration with other information navigation systems.

The functionality of a GAIA broker is rather diverse, and different interactions in the GAIA system present varying requirements to the underpinning technologies. For the purpose of selection of appropriate protocols, the operations of a brokerage system can be broken into the following three categories:

- interactions with the customer
- interactions with other brokers
- interactions with suppliers

The first and last of these occur at the two ends of a supply chain, while inter-broker operations take place at other points in the chain. The supply chain may take a number of different forms:

- a minimal chain, where the customer and the broker are the ends of the chain, and there are no intervening links. In this case, the broker plays the role of supplier to the customer.
- a three-piece chain, where the broker deals with the customer and the supplier, but not with any other broker.
- a longer chain, with one or more inter-broker operations.

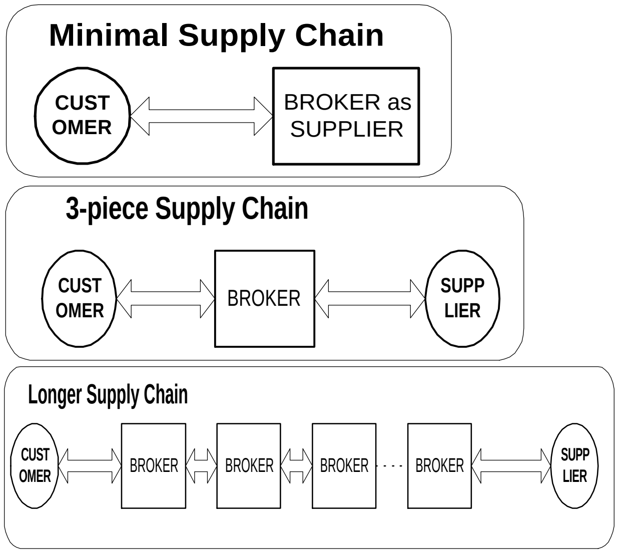

Figure 5. Supply Chains

4.2 Minimal profile

As specified by the GAIA reference model, a GAIA transaction is composed of a number of actions, such as search, order and delivery. Each transaction is initiated by the customer, who makes a request to the broker. In the event that the broker is able to fulfil the request, the transaction involves no other actors. In this simple case, the GAIA transaction involves the customer and the broker, and the only protocol which needs to be standardised is that between the customer and the broker. This is done in a trivial way using the HTTP protocol [11]. The “GAIA standard” refers to such arrangement as a “minimal profile”.

4.3 Basic profile

In the event that the broker is not able to fulfil a request, the action may be propagated on to other brokers, with the original broker playing the customer role. The supply chain is thus made up a single customer, one or more suppliers, and one or more brokers. In order to propagate an action from one broker to another, a standardised communication protocol must be defined for broker-broker interaction. The GAIA Standard prescribes invocation of the remote broker's Customer FU. As shown in Figure 8, several mechanisms for implementing a Customer FU are possible. In order to achieve global interoperability among GAIA brokers, at least one type of customer FU must be made mandatory. Since the interfaces were specified in CORBA IDL, and the overall GAIA architecture matches the distributed object model rather well, CORBA IIOP was nominated as mandatory in the so called "GAIA basic profile". Where two or more GAIA brokers comply with the basic profile, they can be involved in a chained transaction.

4.4 Extension modules

The extension modules in the GAIA standard specify the profiles to be used for various brokerage functions, thereby admitting supplier systems which do not adhere to the GAIA basic profile to the wider GAIA Standard. Example protocols are Z39.50 for discovery, ISO ILL for ordering, and FTP [10] for item delivery. More extensions can be easily added whenever necessary.

Search and *locate* actions involve the customer finding a resource or product with the assistance of the broker. By default, specification of the actions takes place via HTTP [11], with the customer using a WWW interface to the GAIA broker, and the results of the discovery are returned to the customer via the WWW interface. However, if specified, results may also be returned via email, using the SMTP with MIME extensions. It should be noted that searches are particularly likely to be initiated on a GAIA broker by entities which conform only to the Z39.50 protocol in the GAIA Standard. These may be chained using Z39.50 to other non-conformant entities.

Order involves the customer asking that a resource or product be delivered to him by the broker or the supplier. Once again this takes place via a WWW interface, and the order is transported to the broker via HTTP. If the order needs to be propagated, and the basic profile is supported, the order is 'translated' to abstract primitives, encoded with IIOP, which are used for inter-broker communication. At the supplier end of the chain, ISO ILL may be used for ordering from libraries.

The *Delivery* action may either be carried out using a direct supplier-customer delivery (where the broker has only worked as a referral agency, or is itself the supplier), or via a supply chain of multiple linked deliveries. A delivery to the customer may make use of a number of different protocols, including FTP [10], e-mail/MIME [13][14][15], and MPEG/RTP [9][12] (in case of continuous media delivery).

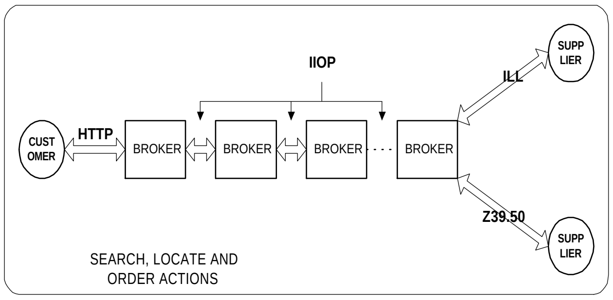

Figure 6. Search, Locate and Order Action Supply Chains

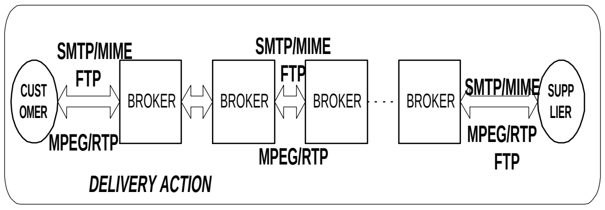

Figure7. Delivery Action Supply Chain

A Broker can choose an appropriate set of Extension Modules to conform to according to the functionality it wishes to achieve. There is one extension module for each of the functional areas which are not covered by the Basic and Minimal Profiles, and one extension module for each of the existing areas (Customer, Discovery and Order) to allow the use of protocols other than IIOP. The extension modules specified and the protocols they are based upon are summarized in Table 2.

Table 2. GAIA Extension Modules

Extension Module	Protocols supported
Item Delivery	FTP, Internet email/MIME, GEDI
Stream Delivery	MPEG/RTP
Security	PKIX / GSS, SSL, Java Access Control System
Payment	SET
Discovery	Z39.50
Order	ISO ILL
Customer	Z39.50, ISO ILL, email

5. Building a CORBA-based brokerage system

As noted earlier, the internal interfaces of the GAIA Functional Architecture were defined in CORBA IDL. This proved to be an excellent choice for a number of reasons. Subsequently the decision was made to also use CORBA as an integration mechanism for the pilot development of the GAIA Broker. This decision too was venerated by experience, although initially there were a number of difficulties.

GAIA made the decision to use IDL for the specification of functional interfaces for the following reasons:

- it is widely accepted as a standard
- it is rigorously defined, independent of platform
- it is easy for systems designers and programmers to learn
- it is readable by technical staff with no CORBA training
- it is supported by tools such as “idldoc” [4] which can produce documentation useful in a collaborative environment.

By May 1997, an initial set of IDL interfaces for the FUMs described above had been produced. In July 1997, the Consolidated Design for the GAIA Broker [2] was published, which detailed the IDL interface of the Broker Kernel, and rationalised the interfaces of the FUMs.

There was some debate as to whether these CORBA-compliant interfaces should lead to a CORBA-based implementation of the GAIA Broker, i.e. with functional integration occurring around an ORB. The alternative was to use lower level protocols to integrate, with higher-level communication provided by a combination of proprietary middleware and GAIA’s own solutions. (Note that Microsoft’s equivalent to CORBA, i.e. DCOM, was not in the running, because it does not yet support cross-platform interoperability)

The early experiments with ORBs were promising but problematic. A key issue was interoperability between different ORBS: prior to CORBA v2.0, no standard for interoperability was defined. Version 2.0 specifies GIOP (General Inter-ORB Protocol) and IIOP (Internet Inter-ORB Protocol) [6], the latter instantiating the former over the Internet Protocol, IP. The first v2.0-compliant ORBs appeared soon after GAIA started work, but interoperability was far from assured.

By December 1997 interoperability via IIOP was established. Systems integration of the basic framework was 4 weeks behind schedule and some “workarounds” were in place to handle some environment problems, e.g. concerning bugs in naming services and IIOP incompatibilities. However, with commercial requirements now mandating the use of a mixed Solaris/NT platform for a distributed Broker architecture, the use of ORBs was felt to have been beneficial. The implementation deployed to demonstrator sites for the GAIA trials makes use of the following interoperating components:

- Broker Kernel, built with Orbix (from IONA Ltd), running on Solaris or NT
- assorted FUMs built with Orbix, running on Solaris or NT
- Directory Services FUM built with OmniBroker (a shareware ORB), running on NT
- Customer FUM interface built with VisiBroker (from Visigenic), running on Solaris

Other ORBs that GAIA worked with include ILU, a very early CORBA implementation from Xerox PARC laboratories, and OmniOrb, another shareware ORB. Of the ORBs tried, only Orbix and VisiBroker gave full v2.0 compatibility, with support for multi-threading and for dynamic service activation. OmniBroker was entirely satisfactory apart from lack of multi-threading, which made it suitable for a singly-threaded systems component such as the Directory Services FUM, but not for the Broker or for the Search, Order or Directory FUMs. Dynamic service activation was considered important for efficiency, rapid start-up response time, and to support service fulfillment by agents.

Overall, the experience of ORB implementation was a good one. All components of the Functional Architecture were implemented with less “compromise” than the project expected. The Kernel and FUMs were built from scratch based on their IDL definitions, and largely coded in C++. The FUs were at least partly based on existing code, with a CORBA wrapper around them to offer the FUM/FU interface.

In the Spring of 1998, the User Interface architecture took shape. A layer of Java classes was built to offer this interface to diverse clients, and two structurally different User Interfaces were produced upon it. The “pure Java” interface comprises Java applets or applications resident or downloaded on the client machine, which call the Broker via IIOP. The “dynamic html” interface uses a proprietary webserver API to link a site-configurable front end to the Java classes resident on the webserver (GAIA used Netscape’s LiveConnect protocol to do this; another choice would have been Microsoft’s ISAPI protocol). In addition, a non-Java User Interface was constructed, which provided access for non-GAIA clients running the Z39.50 search and retrieve protocol [7]: this interface evidently could not support the full range of Broker functions.

As shown in Figure 8, three User Interface architectures were developed for the GAIA demonstrator.

The “pure java” interface is highlighted, as the choice which most readily supports the deployment of brokerage client software in a heterogeneous environment and is the solution being used in the GAIA Technical Components Domain demonstrator. At the time of writing, binding to Java from ORBs is much better supported than binding to C++, Java being amenable to the provision of simpler arrangements for memory allocation etc.

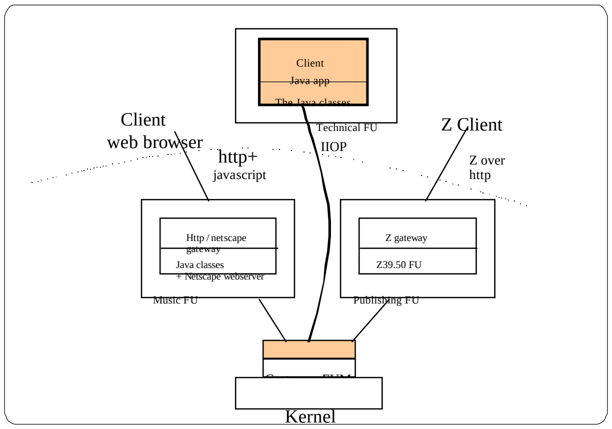

Figure 8. User Interface architectures trialled with the CORBA-based GAIA Broker

Figure 9 concludes the discussion on the GAIA projects experience in developing and implementing a CORBA based brokerage system by presenting an example of the search action as implemented within the Technical Components Domain demonstrator.

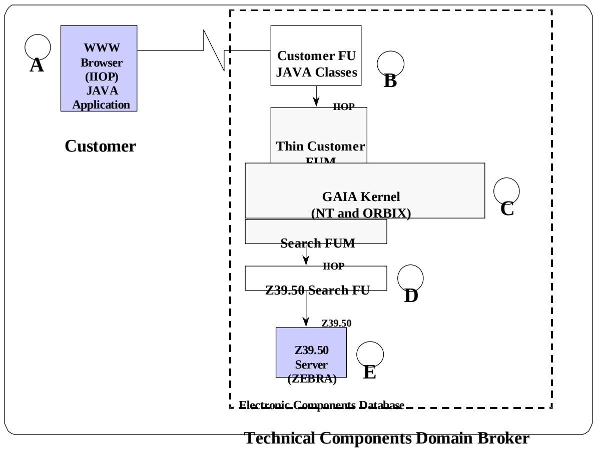

Figure 9: Technical Components Domain Demonstrator

The simplified operation and implementation of each of the shown modules can be summarized as follows:

A. Customer – the GAIA Technical Components Domain Client application is loaded on the customer's Windows based PC. This application has been written in Java and in addition to user interface functions it also includes Interaction Agent functionality [17] to assist the user in the operation of the system. After being authorized by the system the customer performs a search operation by entering his/her search criteria and submitting the request.

B. Customer FU – the GAIA Technical Components Domain Customer FU has been developed with VisiBroker from Visigenic and with Java. Call Back target mechanisms are initiated and used to track the progress of the issued search request.

C. Kernel (with associated FUMs) – the kernel and associated FUMs have been developed in C++ in conjunction with the Orbix CORBA environment from IONA Ltd. The kernel maintains a session for each active GAIA transaction. When the kernel receives the search request for the transaction it will move the session to the search state and pass the request to the Search FUM. In the demonstrator there is only one Search FU implemented so the request is passed to the Z39.50 Search FU.

D. Search FU – the Search FU uses Z39.50 technology to formulate the search request. It was developed using existing software with an IDL CORBA wrapper. The FU will take the search request from the FUM and convert it into Z39.50 format.

E. Supplier Database – the actual technical components supplier data is stored in a ZEBRA database. ZEBRA is a Z39.50 server developed by Index Data IS, a partner in the GAIA consortium. The ZEBRA server processes the search request and returns details of the result set to the Customer via the above described modules.

To sum up the ORB experiences of GAIA, it is noted that cross-platform and cross-ORB interoperability is a reality and that the CORBA approach offers flexibility welcome in the provisioning of services in a distributed environment. Whilst CORBA is at present an immature technology, it can be expected to evolve in the near future and constitute a key enabler technology for Electronic Commerce [18]. Other object-based approaches, such as DCOM, may also become applicable.

GAIA intends to promote its CORBA-based specifications and toolkits. Two areas where CORBA-implementations may provide significant advantages are:

- the Z39.50 search and retrieve protocol
- Directory Services

The advantages of CORBA implementations, as specified by GAIA, are that service creators need not grapple with low-level encoding and transport mechanisms, and can use a much more standardised approach to systems integration based on business-level objects.

6. GAIA's impact on the supply chain

6.1 Customer Impact

The heterogeneous nature of a GAIA compliant broker will ensure that the prospective customer will be able to use their current computer environment and telecommunications infrastructure to access the services of a GAIA compliant broker. Accessing a GAIA compliant broker, as opposed to other current on-line retail points of presence, will ensure that customers can locate, order, and receive delivery of a wide range of products and services from suppliers in a global non-monopolistic environment. The ability of a GAIA compliant broker to add value to the process of a customer obtaining goods and services will optimize the ability of customers to select the right product, at the right price, and at the right time.

6.2 Supplier Impact

As with the customer, the heterogeneous nature of a GAIA compliant broker will ensure that a supplier wishing to promote, sell, and deliver their product or service via a GAIA compliant broker will not have to invest in a new infrastructure. Establishing a commercial relationship with a GAIA compliant broker will allow the supplier to access a larger global potential customer base. Better deals are available via the broker for financial and distribution services than could be negotiated direct with today's service providers. Additionally, the new technologies offered by a GAIA compliant broker allow new functionality to be implemented to aid the supplier. For example, a GAIA broker operating in the office automation domain could use push distribution functionality to automatically distribute, on a try and buy basis, new upgrades to a suppliers word processing application to all customers who have purchased the product via the broker.

6.3 Broker Impact

The adoption of the concepts of the GAIA brokerage environment will create business opportunities for entry into new brokerage markets. The value-added capabilities of the architecture ensure that the perspective broker has the potential to financially prosper from their operation. The generic nature of the architecture allows the broker easily to move into an environment supporting multiple domains. The distributed heterogeneous nature of the architecture will allow the broker to establish business relationships with other GAIA compliant brokers allowing chained brokerage transactions.

6.4 Developer/Implementers

The documentation and toolkits developed as part of the GAIA project will allow implementers of GAIA compliant brokers to easily design and implement GAIA compliant brokers in many diverse domains.

In summary, the adoption of the GAIA supplier-independent generic electronic brokerage concepts, architecture, and standards presented in this paper will allow consumers and suppliers to better capitalize from the supply chain benefits which electronic commerce has to offer.

7. References

[1] *GAIA Project Deliverable D0402* (to be updated as D0403 in November, 98)

[2] *GAIA Project Deliverable D0902* (to be updated as D0903 in December, 98)

[3] Ed. S.Plagemann & J.Hands, "An enterprise model for brokerage", *Guideline 3 from ACTS SIA Chain*, 1997

[4] http://herzberg.ca.sandia.gov/idldoc

[5] http://www.syspace.co.uk/GAIA/

[6] http://www.omg.org

[7] *Z39.50 Protocol Specification*, The Z39.50 Maintenance Agency, http://lcweb.loc.gov/z3950/agency/

[8] Moules "The Middleman Always Knocks Twice", *Information Strategy/Economist Group*, April, 1998

[9] ISO/IEC IS 13818 *Information technology - Coding of moving pictures and associated audio information*

[10] RFC 959 *File Transfer Protocol*

[11] RFC 2068/2069, *HTTP v1.1,* (T. Berners-Lee et al).

[12] RFC 1889, *RTP: A Transport Protocol for Real-Time Applications*

[13] RFC821, *Simple Mail Transfer Protocol*, J.Postel

[14] RFC822, *Standard for the format of ARPA Internet text messages*, D.Crocker

[15] RFC1521, *Multipurpose Internet Mail Extensions Part One:Mechanisms for Specifying and Describing the format of Internet Message Bodies*, N.Borenstein & N. Freed

[16] Smith, Maroulis, "A Transaction Service formulated on an ODP Compliant Engineering Platform", *IFIP/ICCC Proceedings*, Trondheim, Norway, June 1966

[17] Koutsabasis, Darzentas, Spyrou, Darzentas "Facilitating User-System Interaction: the GAIA Interaction Agent", *HICSS Hawaii*, January, 1999

[18] Watson (1998), "Distributed Object Technology, a key enabler for E-Commerce", *TREC*, Hamburg, Germany, June, 1998