Component Integration Technologies for Telecoms Management Systems

V.Wade[1], D.Lewis[2], C.Malbon[2], T.Richardson[3], L.Sorensen[4] , C Stathopoulos[5],

[1]Trinity College Dublin, Dublin 2, Ireland
Vincent.Wade@cs.tcd.ie
[2]University College London, London, UK
{D.Lewis,CMalbon}@cs.ucl.ac.uk
[3]UH Communications, Denmark
lbs@uhc.dk
[4]Systems Technology Solutions UK
sts@anglianet.co.uk
[5]Algosystems, Greece
stathop@algo.com.gr

Abstract.

Because of the cost and complexity of building bespoke service management systems, telecommunications management developers are moving toward the use of off-the-shelf componentware to satisfy their management requirements. However, a crucial problem with such an approach is the ability to integrate components to realise integrated management solutions. This paper identifies the technology requirements for the development of systems which support telecoms management business processes and which are constructed from reusable components. It sets out requirements on component integration technologies that are expected to be key to the development of future operational support systems. The state of the art in relevant component integration technologies are reviewed in the context of these requirements. The paper outlines two telecommunications management case studies which are being conducted to evaluate these integration technologies. The paper draws some conclusions about the relative merits of the different technologies examined and make some suggestions for further work.

Keywords:
Business Process Re-engineering, Component technology, Workflow, Technology Gateways

Introduction

The liberalisation of telecoms markets across Europe and the world has exposed service providers to a high level of competition. This competition is forcing them to reduce costs, improve customer service and rapidly introduce new services. One key way in which these pressures can be addressed is through the improved integration of the many software systems operated by a service provider. This includes amongst others, the integration of different operation support systems.

Component based reuse is seen as an increasingly important software development aid, both within the telecoms industry and in the wider IT community. Building systems from components that interact through well defined interfaces offer a route to reusing software across projects within a telecom system developer and to integrating commodity third party software into the system. Both of these offer development cost savings and improvements in reliability and maintainability. Emerging standards such as Enterprise JavaBeans and CORBA Components are encouraging the development of platforms that directly support component integration. This is prompting the telecoms industry to move towards the widespread adoption of component-based architectures. For example BT and MCI/Worldcom have already published architectural and requirement documents that encourage the migration of their OSS architectures to a component-based platforms. This movement is now also being supported within the TeleManagement Forum. Bellcore has long established its Information Networking Architecture (OSCA/INA) with a similar aim. This has been evolved and validated by the TINA-Consortium, resulting in the standardisation of ODL in the ITU where it is likely to become the basis for the further standardisation of IN Capability Sets. However, many existing architectures, such as TMN, do not directly support component-based systems, and not all the notations and tools currently used in telecoms can fully represent components, e.g. GDMO, UML. The alignment of notations and tools with the modeling constructs and development activities associated with component-based software development must therefore be addressed.

This paper examines the state of the art in component integration technology and assesses it against the requirements of OSS developers. The technologies addressed are:

- CORBA Components: based on the joint revised submission to the corresponding RFP, which is currently under consideration by the OMG.
- Enterprise Java Beans.
- Workflow technologies based on both OMG and WfMC specifications.
- Contemporary TMN technology, specifically gateways between CMIP and other distributed technologies such as CORBA and SNMP, and the C++ TMN API.

Other relevant technologies such as DCOM, Distributed Databases and Transaction Processing have not been covered in this paper but are worthy of further examination in this area. Also identified in this paper is the current TeleManagement Forum's approach to component integration as specified in their Technology Integration Map. The paper concludes by illustrating some of the technology choices and integration achieved in the ACTS FlowThru project which are realising trials in Service provisioning, Accounting and Assurance Management.

Management System Development Requirements

The telecoms industry needs to build solutions to specific management problems from the wide range of architectural and technological approaches currently available from bodies such as the ITU-T, ISO, TM Forum, TINA-C, OMG, ETSI and EURESCOM, among others. In particular, practitioners need to create operational support system solutions from reusable telecommunications management components that may be drawn from multiple origins. Developers of management systems need to be able to make reasoned selections from existing solutions (standardised or otherwise) while ensuring the integrity of the information flows required to satisfy business requirements.

In order to gain a better understanding of the relationships between the various potential actors in the development and operation of services a business model can be constructed, which illustrates the interactions and dependencies between these potential actors. This business model centres on a Management System Developer stakeholder operating in a market where it provides management systems (possibly internally) to a Service Provider stakeholder. A management system must support one or more management tasks required by the Service Provider, which may involve interactions between the Service Provider and Customer stakeholders and/or other service providers. Ideally, the development of management systems should make use of commercial off-the-shelf components, purchased from Component Vendor stakeholders in an open market. The System Developer also relies on the use of open standards for platforms and common management functions. These ensure both the interoperability between the provider's systems and those operated by customers and/or other providers and the interoperability between components purchased from different vendors. The general management system development environment is summarised in Fig. 1.

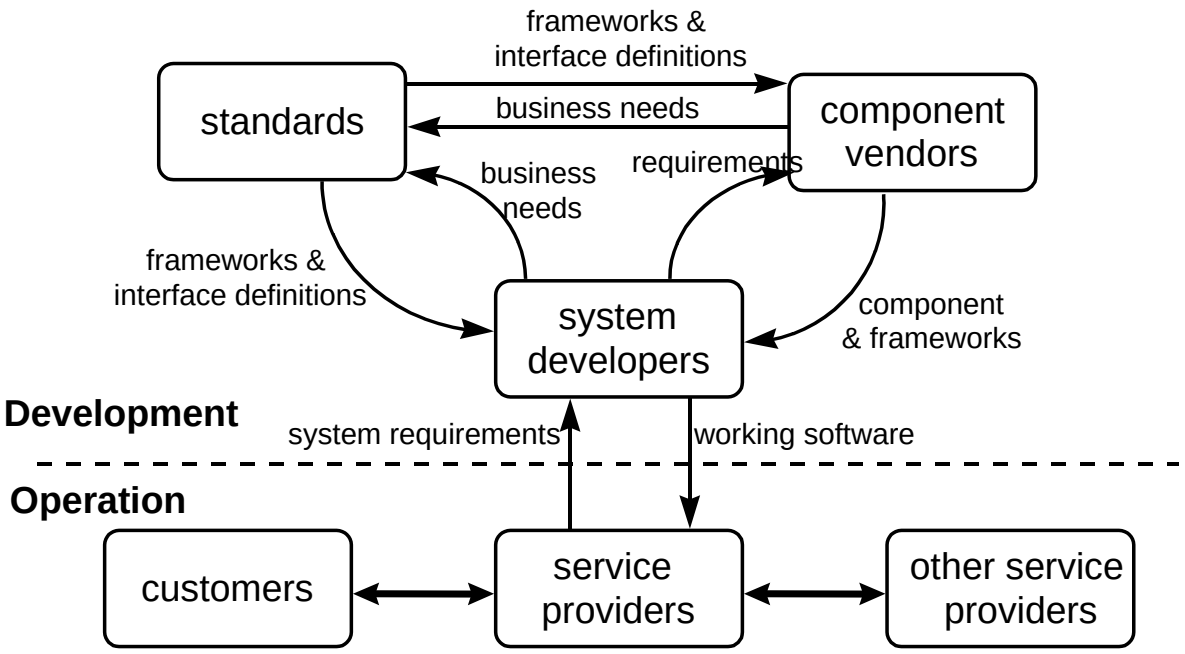

Fig. 1. Business Model for the development of Open Management System

Ultimately we want to support an *open market* in reusable components for management systems, such that construction barriers for building software systems from reusable, multi-threaded components are reduced. Key requirements for such component integration are:

- The ability to integrate with legacy systems in a cost effective way.
- The ability for components to interoperate even if they have been implemented using different distribution or programming technologies.
- The ability for components to interoperate even when they offer interfaces that have been defined in different languages, e.g. IDL, ODL, GDMO or SMI.
- An integration mechanism that minimises the knowledge needed of other potential interoperating components when a new component is developed.
- The need for an integration mechanism that clearly supports the needs of specific business processes in a clearly observable manner.
- The need for an integration mechanism that is robust to changes in technology.
- The need for an integration mechanism that minimises the obstacles to adapting a component to a new application.

TeleManagement Forum Technology Integration Map

The TeleManagemet Forum (formerly the Network Management Forum) undertook and examination of several different integration technologies for telecoms management called the Technology Integration Map (TIM) [1]. The overall structure of the TIM is represented in Fig. 2.

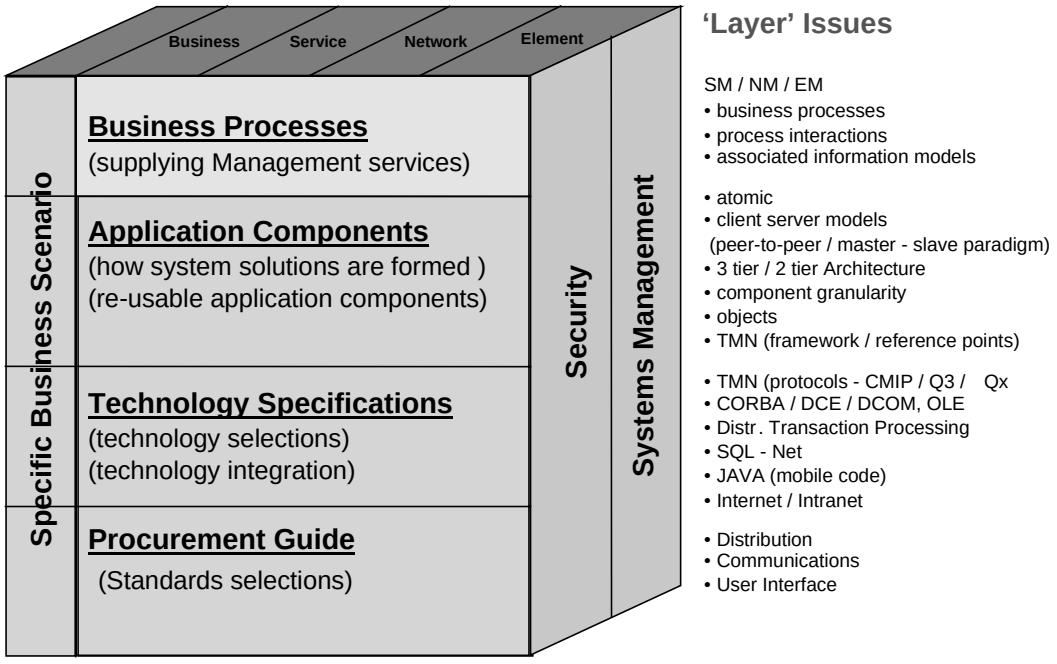

Fig. 1. TM Forum Technology Integration Map Structure

The TIM suggests a possible mapping of technologies and associated application components to particular telecom business needs. The needs of individual service providers are driven by specific business scenarios. The intention of the TMF is to use this mapping for several of its activities, which include:

- Telecoms Operations Map which gives a provider independent view of key business processes
- Application Component activity, which is at an early stage, but which aims to provide the basis for defining reusable management components
- Technology Direction specifying the relative merits of different management-related technologies such as CMIP, CORBA, DCE, DCOM, SQL, Java etc.

- Procurement Guide, which would aim to provide guidance to industry practitioners when making purchasing, decisions for components and platforms of management systems.

Currently the TIM focuses on the comparing the technology specification and does not provide any real guidance on how this might map to *specific business process areas*. It has however recommended where different technologies might be applied to different types of system role.

Interface	Type	Technology
1	Customer/Operational Staff access	Web browser / JAVA
2	Business process interaction / backbone distribution	CORBA (+ Workflow?)
3	Business process control of network resources	CMIP/GDMO SNMP/MIBs
4	Business process access to operational data (not discussed)	SQL, SQL-Net, ODBC, Data Distribution?

Table 1. Technology selections

This categorisation led to the identification in the TIM of several technology integration points, i.e. points where effective interworking were required between different technologies in different system roles that needed to interact as part of an integrated management business process. These integration points (IPs) are identified and numbered (1-5) in **Fig. 3.**

.
.

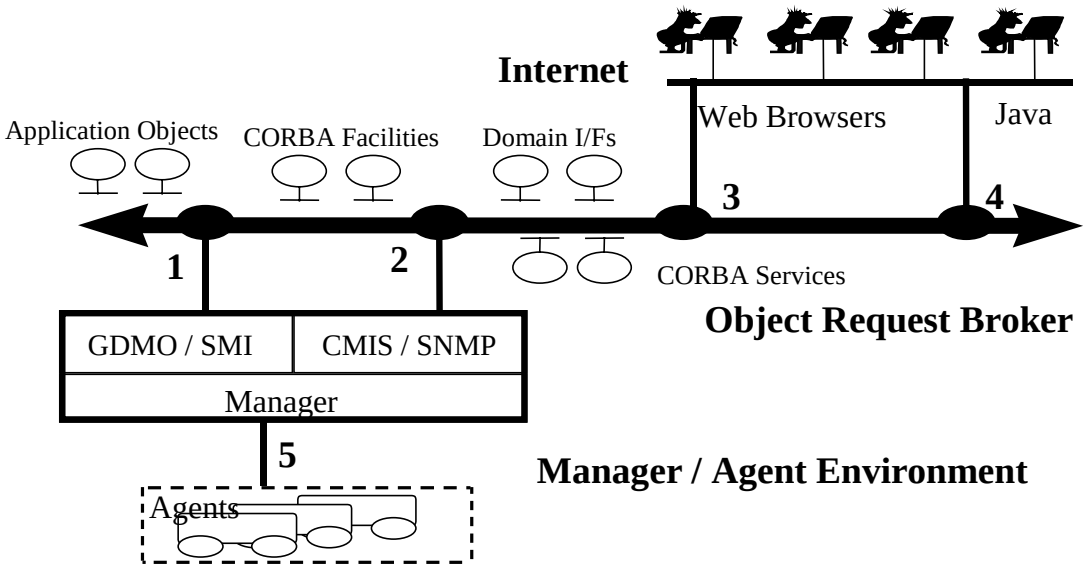

Fig. 3. Technology Integration Points

Essentially, this figure indicates that from the technologies selected, three technology areas will need to be integrated. These are:

- Internet/Web based services
- Object Request Broker (CORBA) based services
- Telecom based Manager/Agent services (i.e. CMIP/GDMO and SNMP/SMI)

In order to provide adequate points of integration between these areas of technology, the five IPs have are identified below:

- **IP1:** Provides mapping of objects defined in CORBA/IDL to managed objects defined in GDMO or SMI.
- **IP2:** Provides mapping of appropriates CORBA Services to CMIS and SNMP services.
- **IP3:** Provides a mapping of Web Browser technology access to CORBA objects (for situations where this may be needed as an addition to/replacement of Browser access to a database).
- **IP4:** Provides a mapping between Java based objects and CORBA objects.

– **IP5:** Provides a high level convenient programming interface for the rapid development of TMN based manager/agent interactions. It also provides a convenient point of integration if it is necessary to separate out the two sides of the manager/agent interface from the point of view of technology selection. For example, allowing the manager role to perhaps be supported in a Web-based environment, but giving a good point of integration with a TMN based agent.

While some of these reference points have been addressed by existing gateway standards, e.g. the JIDM CORBA-CMIP specifications, key technologies such as Java, CORBA and Workflow have been rapidly evolving towards direct support for reusable components in a way not currently addressed by the TIM. The following sections review some of the relevant interworking technologies and the component oriented developments that have emerged recently.

Management Component Integration Technologies

This section will discuss three integration approaches, namely TMN API, CORBA / TMN interworking, Component Technology (e.g. CORBA Component Model and Enterprise Java Beans), and Workflow.

TMN Integration

In the context of this paper TMN means the OSI management framework as standardised in the ITU-T X.700 series of recommendations. The definition of client/server relationship between managers and agents are modelled in GDMO/ASN.1, and the actual exchange of management information between systems is via the CMIP protocol.

The interoperability between TMN based systems is good. Although CMIP is fairly complex, the experience shows that it is most often possible to configure systems to interoperate at the protocol level (as opposed to some CORBA based systems). TMN based management systems have great potential in many aspects. If this is true then why is TMN based systems then not deployed more widely? Probably because early TMN systems got the reputation of being expensive and difficult to implement. However, this does not imply that TMN has no future. The difficulties in implementation are reducing considerable, due to new and highly efficient development toolkits. It can be argued that because of TMN's maturity, its technology has reached a stage where development of a TMN based management system is less demanding in terms of implementation effort compared to any other technology. This is largely due to built-in generic and standardised features that often are needed when building management systems. A modern TMN development platform offers event handling, distribution, naming, persistency, logging, design tools, debugging, test facilities and a number of implemented standardised information models even before the implementation work is started.

The TeleManagement Forum has, with the TMN/C++ Application Programming Interface (API) product set, defined a standard programming interface for use with these international standards for network and systems management. The objective of the TMN/C++ API is to provide a straightforward mechanism to write portable and interoperable management application programs using C++.

To support this objective the TMN API has been designed to:

- Allow widespread portability, with a minimum amount of effort and code modification.
- Be simple, intuitive, and easy to use for experienced programmers.
- Be sufficient to support anticipated applications.
- Be flexible to allow for different implementation strategies, application designs, and operating environments.
- Be useable by a wide variety of application problem domains (i.e. it should be neutral with respect to the actual GDMO content.)

Overall the above features result in application code that has a high degree of reusability between applications and even between implementation platforms.

The concept of Q-adaptation paves the way to integrate legacy systems with TMN based management systems. The specification translation between GDMO, SNMP and IDL(CORBA), as defined by TMF, makes it possible to automate the building of gateways between CMIP, SNMP and CORBA. The gateways ensure that components based on these technologies can work together, and implementations have already proven that the concepts are feasible.

CORBA / TMN Interworking The Joint Interdomain Management (JIDM) task force was set up by TeleManagement Forum TMF and X/Open to provide specifications of how CORBA and OSI System Management (SM) could co-exist in the same management environment. Their specifications involve translating information exchange between TMN- and CORBA-based components. There are two streams in this work –

mapping between GDMO/ASN.1 and CORBA IDL, and translation between CMIP/SNMP and CORBA operations invocations. The former consists of facilitating information models in either domain to be viewed by its counterpart. The latter is concerned with component interactions in each domain – CORBA and TMN – providing the capability for components in one domain to act like components in its counterpart's domain. Many descriptions of the work and results of JIDM are available e.g. [2]

It also should be noted that, at the time this version of the Technology Direction Statement was produced, the OMG had partially completed the process of adopting a technology for CORBA/TMN Interworking. It also should be noted that the principal submissions to this process were centered on the JIDM and TMN C++ API specifications outlined above. Hence the outcome of the OMG's selection process could well result in a CORBA/TMN interworking specification which is agreed by TM Forum/Open Group and OMG. This would result in greater clarity in the defined roles for both CORBA and CMIP/GDMO, and would see both of these technologies being more fully exploited in future telecom systems.

Component Technology

The advent of component-based development promises to reduce system and application development to a process of wiring together software "integrated circuit" [3]. To achieve this a component must be able to offer both structural and promotional interfaces. Promotional interfaces allow system developers and third parties to inspect a component in order to determine its usefulness. Structural interfaces allow application developers, integrators and frameworks the means with which they can establish communication.

One key benefit that component technologies promise is software reuse. But why should this be any more attractive than with existing OO development methodologies? The answer lies on the shift of the software development process from a craft-based activity (i.e. building from scratch) to an industrial process. Here, component-based systems reduce the application and system development process to one more familiar in traditional industrial manufacturing. Here, large cost savings are achieved by using standard procedures to assemble complex products from sets of well designed, pre-fabricated parts without a loss of build quality. This is sometimes referred to as the "industrialisation" of the software development process. Such a shift requires a big change in software developer mentality. Existing component-based development techniques are related to well-known OO development practices (such as, business analysis and use case modelling). These will not disappear, but, the deployment of component built systems will shift the emphasis from fabrication to integration, and to this end a component's self-description, or "introspection", will become as important as its ability to implement some set task.

A number of emerging heavyweight technologies are addressing component design and development including Microsoft's Active X Controls/DCOM, Javasoft's Enterprise JavaBeans [4] and the OMG's CORBA Component Model [5]. The following sections outline the last two of these technologies.

CORBA Components As a component can be viewed from several different standpoints depending on how the component is being used and who is viewing the component the CORBA component specification defines a number of separate models through which a component must be described.

The Abstract Model describes a component from a top-level, design perspective. Also known as the Meta-Model, it contains information on the objects, links, and data values of the component, as well as details on how to this information should be viewed using Document Type Definitions (DTDs) based on the XML Metadata Interchange (XMI).

At a lower level, the specification defines five further models that describe how the component will both function and integrate. These are the Component, Persistency, Packaging, Deployment and Container models.

The Component Model expresses the component as a type. The type definition provides both compile-time and run-time information on its external interfaces. This includes new IDL syntax to provide:

- Unique component identification
- Identification of interfaces that the components both provides and uses
- Details on the events that a component both emits and consumes
- Navigation interfaces that allow the above to examined
- Interfaces that allow runtime attribute and property configuration
- Interfaces for managing multiple component instances

The new syntax is essential to allow an IDL compiler to turn a component definition into an executable implementation in a target language.

The Persistency model defines how a component's state may survive its physical representation in memory, extending a component's lifecycle.

The Packaging Model provides details to enable a component to be assembled with other components within the scope of a distributed application.

The Deployment Model defines how components are configured and readied at run-time. The main aim here is to install the component into a physical computing environment.

Finally, the Container Model deals with a component's runtime environment. In general, components execute in containers and must include details of the system services they require in order to operate effectively. Its container, itself a component, provides access to these services and controls external access to potentially multiple instantiations of the component.

Enterprise Java Beans SUN Microsystems's Enterprise JavaBeans (EJB) is designed to run within the Java Runtime Environment and therefore requires a suitably engineered Java Virtual Machine. In order that an EJB may be queried by frameworks and other EJBs and connected to those that wish to cooperate, it must provide a number of standard, external entry points. The specification defines a set of naming conventions, "design patterns", that enable a component's entry points to be identified and which builds on the introspection inherent all Java classes. These entry points include: a set of methods or interfaces via which the component can communicate; run-time configurable properties that promote customisation; and events with which a component notifies interested EJBs of changes in state.

Crucially, a JavaBean executes within a container, in itself a JavaBean. A container provides an environment within which one or more beans run and acts as a bean coordinator, controlling access to the bean and creating new instances on request. As the EJB spec, "A container is where an enterprise Bean object lives, just as a record lives in a database, and a file or directory lives in a file system". This enables beans to be managed in a locally scalable way. It can also control bean access to supporting services and scarce resources, and provides a secure environment for each bean to exist within the wider distributed environment.

To facilitate this an EJB exists in one of two different guises, as either a *session* or *entity* bean. Session EJBs are transient and have a limited lifecycle. They are created by clients and usually only exist for the duration of a client/server session. Once the bean is destroyed its data is lost, which, crucially applies in the event of a system crash. The bean must manage any persistency that might exist. On the other hand, an entity EJB is persistent and typically lives as long as the data that describes it. As the data may reside in a database this could be for a much longer time than the components initial representation in system memory. Furthermore, EJBs can survive a server crash and are designed to be transactional.

To complement the EJB distributed component model SUN has defined a comprehensive set of APIs. There are 8 in total and consist of the: Java Naming and Directory Services (JNDI), Remote Method Invocation (RMI), Java Interface Definition Language (Java IDL), Servlets and Java Server Pages (JSP), Java Messaging Service (JMS), Java Transaction API (JTA), Java Transaction Service (JTS) and Java Database Connectivity (JDBC). Together these form the Java Platform for the Enterprise (JPE) and this is intended to enable any Java application to access an enterprise-class infrastructure, often termed middleware.

CORBA Components and EJBs, CORBA Component and EJB specification are alternative component technologies. The EJB has already presented one of the first working implementations, backed by the Java market. The CORBA Components present a more general modelling framework, advocating the use of CORBA as the underlying technology. The forthcoming CORBA Components standard will constitute an integral part of the third version of the CORBA suite. It will contain both the models and the meta-modelling constructs required in order to deploy a full-scale component technology. Additionally, it will cover the mapping from Enterprise JavaBeans (EJB) components to CORBA components and vice versa. In fact, one of the mandatory requirements set by OMG, in its Request for Proposals for a CORBA Component Model [6], was the compatibility with JavaBeans, while the latest proposal has considered compatibility with Enterprise JavaBeans. It is expected that an Enterprise JavaBean may be inserted into a CORBA container, while a CORBA Component, written in Java, may run in an EJB Container. An EJB interface can be mapped to its equivalent CORBA IDL and an EJB container can use either RMI or IIOP as its distributed communication protocol. More specifically, an EJB may be inserted into a CORBA container and a CORBA component, written in Java, may run in an EJB container. Furthermore, certain JPE services have equivalent CORBAservice counterparts from which they borrow much of their design. JTS is an implementation of the CORBA Object Transaction Service, and the JNDI extends the CORBA Naming Service to give general directory access support that includes the Lightweight Directory Access Protocol (LDAP).

Workflow

There is a growing awareness that workflow management tools and techniques could provide a vital element in the co-ordination of distributed components within different provider domains whilst allowing greater flexibility and the necessary degree of autonomy [7]. Because of the introduction of new services, new relationships with other service providers or new functionality or equipment, the management components (form service provider and network provider systems) must be capable of adapting rapidly to changes in the way the business process is executed.

A workflow management systems is a system that defines, manages and executes workflow processes through the execution of software whose order of execution is driven by a computer representation of the workflow process logic. Workflow technology incorporates the benefits of co-operative information systems, computer-supported co-operative work, GroupWare systems, and active databases. Workflow management technology addresses the following requirements:

- Improved efficiency, leading to lower costs or higher workload capacity
- Improved control, resulting from standardisation of processes
- Improved ability to manage processes; identification and analysis of problems
- Cost reductions, where cost can be a euphemism for staff
- Increased quality or capacity while controlling costs
- Construction of unique customised business processes to deal with specialised management work practices
- Improved information distribution, and elimination of the delays caused by the need to move hard copy information around the organisation
- Reduced bureaucracy improved the quality of work, decreased cycle times, and acquisition of better management information about business processes.

Thus workflow management can be considered a very attractive technology for integration and interrelation of telecommunication management components. The purpose of applying workflow management technologies in the service management problem domain is to integrate and re-purpose management components that resolve telecommunication business problems, and to automate telecommunication management services. Such enhancements, i.e. the interrelation of service management components, reduce the business process complexity, improve resilience, and improve the overall performance of the network operators' business process.

There are many differences between the architectures, which are used by workflow systems. However most of the workflow systems fall into one of two broad categories [8]:

- Forms and messages based workflow systems which performs electronic routing of forms to user's e-mail in-boxes
- Engine based workflow system, which communicates with humans or components via specialised client software.

It is the workflow engine based approach, which we will focus on to achieve management component integration.

Workflow Engines A Workflow Management System (WFMS), as defined by the Workflow Management Coalition, is a system that defines, creates, and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, invoke applications (or components) [9]. A workflow engine is the basic workflow management control software. This software is often distributed across a number of computer platforms to cope with processes, which operate over a wide geographic basis

The workflow engine controls the flow of work (sequences of management activities which form a management business process) through the system by interpreting the management process rules to determine the scheduling of required activities, and invoking the relevant management components. The engine is responsible for:

- Business process creation, deletion, and management of process execution from instantiation through completion
- Control of the activity scheduling within an operational (business) process.
- Interaction with management components and/or human resources (which execute the required management activities).
- Monitoring and control of the management processes in execution.

Figure 4 illustrates a generalised workflow engine which accepts a (management service) request and based on its process rulebase, invokes the correct sequence of components. The components pass application specific values (input and output parameters) via the shared (component) data server.

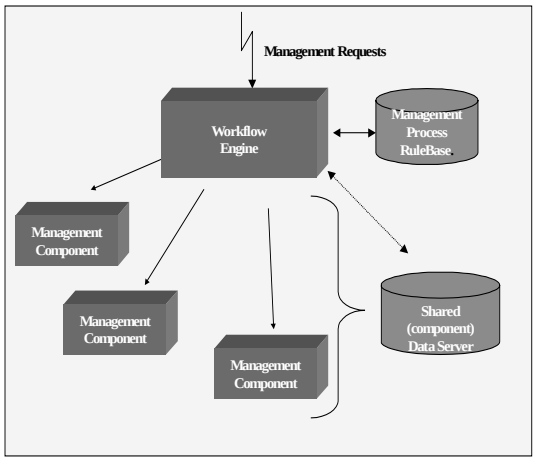

Fig. 4. General Workflow Architecture

There are a great many products and research projects which offer support for workflow management. In April 1996, 250 products claimed to support workflow features and/or workflow management[10]. This constituted a market size of more than one billion dollars. The number of products has risen since then.

Many of the products simply provided a means of graphically representing a business process using techniques such as dataflow, digraph, flowchart, network, orgcharts, pertcharts etc. e.g. Zippen [11]. Others are data management systems, which use e-mail, imaging, databases, electronic forms, engineering drawings etc. to collaboratively process documents or data. Groupware also forms part of this group, Lotus Notes being a good example.

All of these systems have an emphasis on office processes, e.g. imaging, document routing, enhanced mail. However a number of limitations are evident with these types of workflow systems [12].

1. Lack of support for heterogeneous computer systems
2. Incompatibility between workflow products
3. Failure to capture distributed/true nature of infrastructure in business model
4. Scalability not achieved
5. Very little support given towards fault-tolerance and reliability.

Most of these products were designed for small collaborative projects with small loads. As such, they are unsuitable for large-scale workflow management, potentially involving several thousand users, hundreds of thousands of concurrently running processes and several thousand sites distributed over wide area networks [13]. Research projects, however, have confronted many of these issues. Many current research projects are drawing from technologies such as objects, the World Wide Web, CORBA, transaction processing, Java and others in order to help solve some of the problems mentioned above.

CORBA and Workflow, CORBA is used in varying degrees within many workflow research prototypes. In the simplest case, CORBA is used for database access or as a wrapper around legacy applications. The Mentor project uses CORBA to provide a uniform interface to heterogeneous invocable applications [14]. The WebFlow project has a CORBA based CLF co-ordinator which communicates with legacy applications, a relational database and a document management system through CORBA [15].

The WorkWeb project uses a network of CORBA-based agents, where each agent represents a resource or a participant[16]. These agents collaborate and vie for resources. In OrbWork, again based on METEOR2, transactional concepts are implemented using CORBA to achieve fault tolerance[17]. It includes a layer of CORBA-based system components and failure detection mechanisms, which increase availability and allow recovery. Persistence and scalability are other key requirements, which led to the use of CORBA.

The OrbWork project is a CORBA-based workflow engine which contains a ‘Workflow Model Repository’, a task manager (combining the duties of the scheduler and dispatcher), a monitor (holding state for the system as a whole) and tasks (wrappers around legacy applications).

Workflow Standardisation The standardisation of workflow systems has been on-going since 1993 with the formation of the Workflow Management Coalition WfMC (an industrial consortium which set about standardising an architecture for workflow engine based systems, and several interfaces for application invocation, process definition, process management and system interoperability. In 1998, the OMG ratified the definition of a workflow facility, which was based on the WfMC standards

Case Studies – Integration of TINA based Telecommunications Management Architecture Components

The ACTS FlowThru project is currently evaluating the component integration technologies mentioned above, across a range of TMF related business processes, namely Service Fulfilment, Service Accounting and Service Assurance. Two such case studies are sketched below. The first, looking at Service Fulfilment uses CORBA component technology can be equally applied to Enterprise JavaBeans due to their close similarities. It is the subject of a yet to be published paper which presents the analysis, architecture and design of a system that integrates service and network management building blocks in order to satisfy business process requirements for service order fulfilment [18]. The second looks at Service Accounting and uses a workflow engine to integrate a range of accounting components.

Case Study 1 –CORBA Component Integration

Briefly, this takes a number of existing TINA compliant service and network management components, themselves the products of previous projects and uses proposals set out in the CORBA Components specification [5] to produce components that satisfy the TMF FulFilment business process. The scenario relates to the fulfilment of customer orders for a switched ATM service.

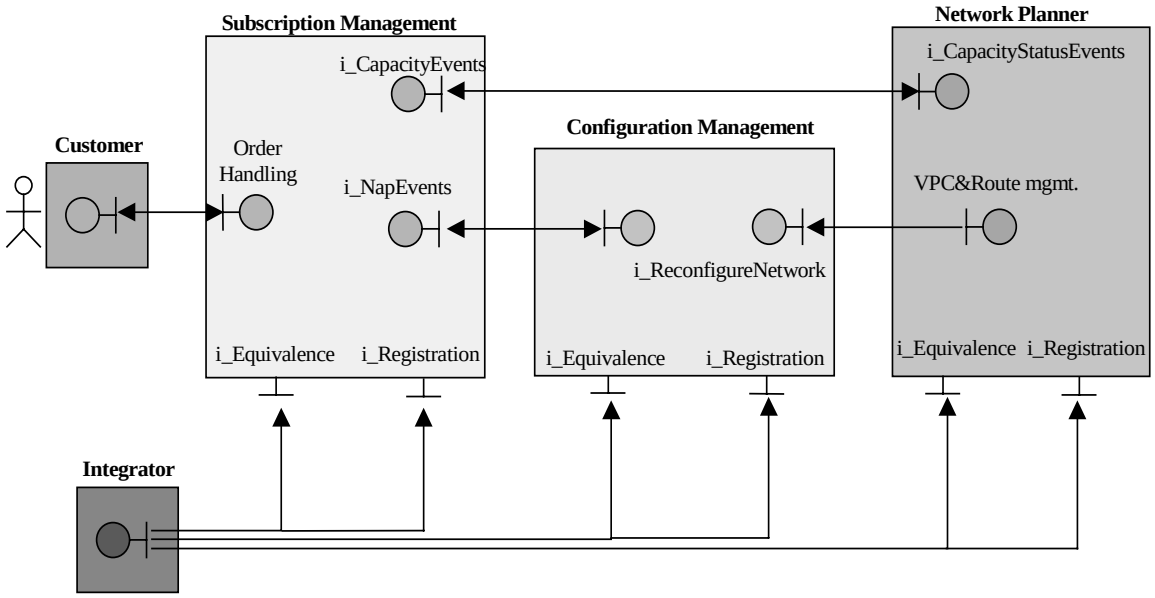

Fig. 5. TINA based Service and Network Architecture integration using CORBA Components

The three principle components: Subscription Management, Network Planning, and Network Provisioning are used as off-the-shelf components. An 'publish-subscribe' eventing model is constructed around each component to provide inter-component asynchronous event communication. The eventing model draws heavily on the Observer design pattern espoused in [19] and the EJB Delegation Event Model. Each component provides standardised introspection and registration interfaces through which a "framework" integrator object can extract their consumed and produced event interfaces. A component desire to receive, or consume, certain events is matched by the integrator object and the relevant publisher informed through a registration interface (Fig. 5). Once registered events are 'pushed' from publisher to consumer, responses where needed are managed similarly establishing an 'event protocol'. Other components can be introduced seamlessly to the system and are similarly integrated so long as they can be matched with a cooperating component. Decoupling components in this way encourages reuse as existing components can be adapted to take advantage of the Fulfilment event protocol. New protocols can be developed to extend functionality, as the integrator object will match cooperating components.

Case Study – Integrating Telecommunications Management using Workflow Engine, and JIDM compliant CORBA/CMIP gateway

The ACTS FlowThru project is currently evaluating the component integration technologies mentioned above, across a range of TMF related business processes, namely Service Fulfillment, Service Accounting and Service Assurance. In the area of Service Accounting, FlowThru is integrating a TINA compliant service management platform with a workflow engine. The workflow engine is used to integrate a range of accounting components e.g. account manager, tariffs Control, Bill Control, Charge Control, User Metering Management etc. Figure 5 outlines the TINA architecture and the workflow engine. The scenario depicted is that for a customer accessing a TINA based Service Management system (Accounting Session Component). The service is being offered by a service operator how is independent of the ATM network operator. The ATM network is managed using TMN technology and ATM usage information is compiled into Call Data Records and emitted as charging notifications. They are received by a JIDM compliant CORBA/CMIP gateway (in the service operators domain) and delivered to the service operators accounting system in CORBA format, where charges for each customer for ATM usage is extracted. The service management system has interfaces into the Subscription Component and Accounting Component. The Accounting Component is actually made up of a WorkflowEntry Agent, the workflow engine and several accounting specific components. It is the workflow engine that co-ordinates the interactions and sequences of these accounting specific components to ensure the desired management activities are carried out e.g. generate a bill for the customer, generate tariff information, provide account management controls etc. All component interfaces are specified in CORBA IDL and components execute of a CORBA based infrastructure..

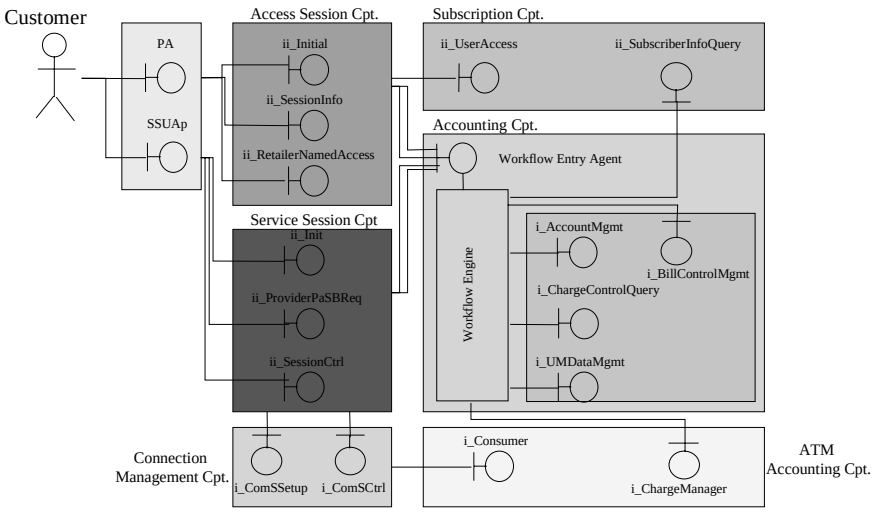

Fig.6. TINA based Service architecture using Workflow engine based Accounting Component Integration

The use of the engine for the accounting component allows the easy introduction of new accounting components without disruption or changes to other accounting components. On introducing a new component into the accounting system, the operator needs to either compose new or amend existing (accounting management) business process rules. This composition/amendments are required as the engine uses these business rules to decide which accounting components to invoke to satisfy a management request.

Conclusion

This paper has outlined the context in which telecommunication management components would be developed and operated. The irresistible move towards componentisation of the management systems has been discussed and the current TMF position on component technology has been described. The paper has also discussed four key integration technologies, namely TMN API, TMN/CORBA interworking, Component Technologies from OMG and SUN, and Workflow. The paper also presented two brief illustration of how these integration

technologies can be deployed to support componentised service and network management systems. These illustrations are based on the TMF business processes of Service (Billing) Accounting and Service Fulfilment.

The CORBA Component case study illustrates how legacy components can be seamlessly integrated using introspection interfaces and a ‘publish-subscribe’ event model. Asynchronous event communication, although more complex from a design point of view, decouples components so that additional componentry can be introduced, for example event logging purposes, without impacting current functionality. Similarly existing functionality can be adapted and extended by introducing new event protocols. Introspection foresees the introduction of component frameworks responsible for connecting cooperating components, here simulated by our integrator object. The model can be easily extended to provide property interfaces that would allow components to be configured ‘on-the-fly’.

With regard to the Workflow case study, the use of the engine had several positive effects on the overall design and implementation. Firstly, it was possible to directly map the accounting business processes into the workflow engine rulebase. This provided a means of validating the system behaviour at a high level. The engine also facilitated the ease of introduction of new components as well as the composition of new business processes based on the existing accounting components. Research into the workflow engine is ongoing to provide improved (management) business process management. Research is also ongoing to provide tools for management service creation based on the workflow engine integration.

References

[1] NMF Technology Map, Draft NMF GB909, July 1998

[2] http:\\WWW.TMF.ORG

[3] Cox, B., Novobilski, A., Object-Oriented Programming : An Evolutionary Approach, Addison-Wesley, May 1991

[4] Matena, V., Hapner, M., Sun Microsystems: Enterprise JavaBeans, Version 1.0, 21 March 1998

[5] CORBA Components: Joint Revised Submission. OMG TC Document orbos/99-02-05, Draft, March 1999.

[6] RFP for a CORBA Component Model. OMG TC Document orbos/97-06-12, July 1997.

[7] M Rusinkiewicz, A Helal "Workflow Management Systems", published in Journal of Intelligent Information Systems, Vol 10, Number 2, March/April 1998

[8]"Ovum Evaluates: Workflow" - Heather Stark, Laurent Lachal, ISBN 1898972605, Ovum, 95.

[9] Workflow Management Coalition Homepage http://www.wfmc.org/

[10] Amit Sheth, "State of the Art of Commercial Technology and Research in Workflow Management" DARPA/ISO Workshop on collective Action Tools - April 10, 1996,

[11] Zippin Project Homepages, Stefania Castellani, Xerox Grenoble, http://www.rxrc.xerox.com/research/ct/prototypes/aippin/home.html

[12] Amit Sheth, "From Contemporary Workflow Process Automation to Adaptive and Dynamic Work Activity Coordination and Collaboration", University of Georgia SIGGROUP Bulletin, Vol. 18, No 3 (December 1997)

[13] G Alonso, D Agrawal, A El Abbadi, C Mohan, "Functionality and Limitations of Current Workflow Management Systems", IEEE Expert, Vol 12, No. 5, Spet/Oct 1997

[14]J Weissenfels, D Wodtke, G Weikum, A Kotz Dittrich, http://paris.cs.uni-sb.de/public_html/papers/mentor.html

[15] Antonietta Grasso, Jean-Luc Meunier, Xerox, WebFlow Homepage, http://www.xrce.com/research/ct/prototypes/webflow/home.html

[16] H Tarumi, K Kida, Y Ishiguro, K Yoshifu, T Asakura, "WorkWeb Systems - Multi Workflow Management in a Multi Agent System", SIGGROUP 1997, Pheonix, Arixona, USA

[17] S Das, K Kochut, J Miller, A Sheth, D Worah, "ORBWork: A Reliable Distributed CORBA-Based Workflow Enactment System for METEOR2", Technical Report #UGA-CA-TR-97-001, Department of Computer Science, University of Georgia, Feb. 1997

[18] Lewis, D., Pavlou, G., Malbon, C., Stathopoulos, C., Jaen Villoldo, E., “Component-Based Integration of Customer Subscription Management with Network Planning and Network Provisioning”

[19.] Gamma, E., Helm, R., Johnson, R., Vlissides, J., Design Patterns : Elements of Reusable Object-Oriented Software, Addison-Wesley, October 1995