# Integrating OSS Components through Contract-defined Interfaces

J S Archer†

† University College London

**Abstract:** As network operators seek to offer niche services and reduce time-to-market there is increased pressure on management systems to respond quickly and flexibly. They need to be able to integrate with the business systems of new partners from outside our industry and change business models in response to market conditions. This paper presents a technology-neutral component architecture which will drive integration from high-level business policies and processes.

## 1 Introduction

With high levels of debt and falling share value both fixed and mobile network operators are reluctant to invest in capital equipment at the present time. Much functionality is already duplicated and specific to particular services. Often end-to-end integration crosses organisational boundaries making it difficult to agree responsibilities and priorities, the more this can be automated the faster integration can occur. Far too much information is presented to Network Operations Centres to be useful and it is not related to the potential impact on the business. In the first instance, network operators are looking to simplify their exisiting infrastructure and reuse it for new services but the ultimate aim is to drive system integration from ad-hoc high-level processes which have a business context.

The New Generation Operational Support Systems (NGOSS) Architecture [1] from the TeleManagement Forum is intended to replace the TMN, eventually. An NGOSS is built from a loosely-coupled selection of components or building blocks. It comprises business and framework services, a common communications service, contract interfaces and a common information model. It has an external registry which contains the location and state of all available software services, the location and state of all computing nodes in the system configuration, the identification and state of all authorised users of the system.

Business services are offered through contract specifications and are supported by basic framework services. Framework services are divided into those that are mostly concerned with supporting the needs of a distributed system and those that more directly support business services. They include communications, invocation, adaptation, security, process management, transaction support, policy management, shared information management, naming, directory, registry and trading (transparency).

The three main principles of NGOSS are:

- to externalise the business logic
- trade contract-defined interfaces
- share common data and/or data definitions (metadata) between components

## 2 Externalising Business Logic

When business processes need to be altered it is often the case that the application components themselves need to be changed as some part of the process is embedded in them. This does not make it easy to understand the implications of events in the network on the business's bottom line. For example, if I wanted to differentiate my response to a problem based on customers' loyalty or usage volume I would need to undertake a considerable amount of re-engineering of existing systems. By abstracting the business process flow into a dedicated framework component it becomes easier to create new and change existing business processes through modifying the configuration of components as opposed to changing the applications themselves. A component simply performs the service offered through its interface, as represented by a contract, and it is then easier to identify suitable candidates for reuse.

### 2.1 Business Process Engine

The Business Process Engine could be a simple scheduler or workflow manager. But ideally it should query a registory to discover what 'contracts' exist that can fulfil a particular business process. It is not yet clear how business policies will interact with the process. The implementation below [Fig.1] used the Policy Engine from Telcordia to determine the best configuration based on the available resources and other context information.

### 2.2 Business Transaction Patterns

The UN/CEFACT Modeling Methodology (UMM) [2] defines six business transaction patterns (such as request/response). These do not imply any particular implementation but instead define what high-level behaviour must arise out of the collaboration between businesses, which at a systems level must be translated into relationships between functional components. The business transaction approach to contracts using bilateral collaboration patterns provides for a state-wise process integration, not just message exchange for data sharing. But the question remains, where in the NGOSS architecture will global state be visible? A suitable candidate is a Business Process/Policy Engine framework component at a level of abstraction between high-level business processes and their mapping into system-level processes.

## 3 Contract-defined Interfaces

Applications will locate each other and discover what services they offer through a repository of contracts. A contract is a well-formed interface by which the functionality of a component is made available to a client. Components advertise their contract instances by their class and the values of their service attributes, via a trading service. The remainder of the contract comprises the operations by which it provides the system services for which it was designed.

Contracts comprise a technology-neutral portion and a technology-specific portion. The technology-neutral portion is independent of the protocols used between components, whereas this needs to be specified in the technology-specific portion which will be used at the systems level. So the interface specification is independent of the communications infrastructure.

It is possible that the technology-specific portion will not be completed until runtime. The reason for this is that the introduction of a new component can trigger the reconfiguration of the NGOSS adapters of existing components and consequently alter the communications between components. This enables a new component to enhance an existing business process (Fig.1), dynamically. This is not suitable for long-lived assurance scenarios.

There is much discussion about whether the technology-neutral portion of contracts should contain pre- and post-conditions and what the implications would be. If commercial off-the-shelf components did not coincide closely with the functional areas defined by the SIM then specifying conditions in this way could be exploited to undo the decoupling that the architecture achieves in other ways. And there is also a danger that state information is distributed around the architecture making it difficult to ensure that adverse service interactions are not taking place.

## 4 Common Communication Infrastructure

NGOSS has its roots in the 'plug-and-play' philosophy. The main concern of plug-and-play systems is to be able to easily swap out legacy business components in order to swap in state-of-the-art replacements. But this necessitated defining functional blocks as in the Systems Integration Map (SIM) [3] which would have forced vendors to compete directly on well-bounded functional grounds, which they are understandably reluctant to do. But another aspect of plug-and-play systems is their scaleability.

Many operators are constrained by particular systems which are on the critical path for the introduction of most new services. They work but they are stretched to their limit. Billing mediation is a classic example. Such hubs of activity are a mesh of inter-dependent systems; change one and you are likely to need to change several. A common communication infrastructure is intended to reduce an $n^2$ problem into an n problem, that is to say only the interface to the common communication infrastructure, for example a bus, need be changed.

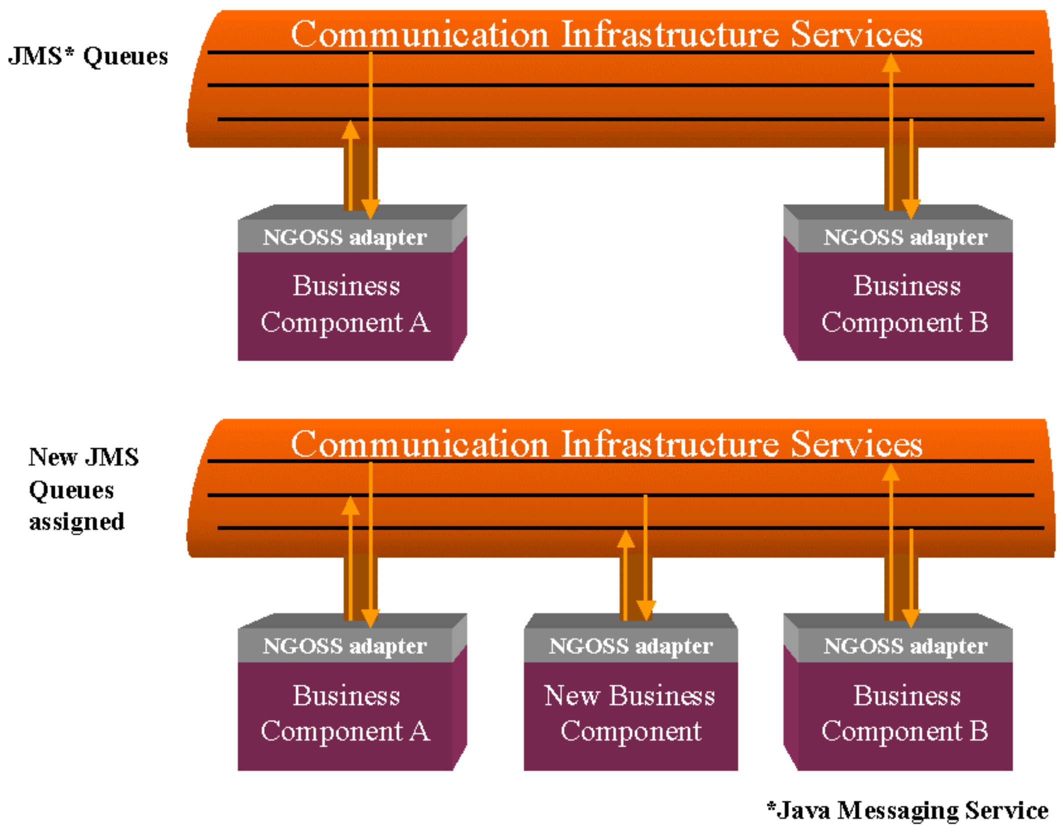

**Fig.1 Using contracts to change the communication flows at runtime**

This does not imply that all communication between components must go across the common communications infrastructure. On the contrary, this would present a single point of operational failure. The common communications infrastructure is intended to facilitate component discovery and configuration upon setup of business processes. It also monitors resource capacity and availability.

## 4.1 Common Information Model

The key aspect of any common communication infrastructure is a common information model. To date, the definition of information constructs to support the interchange of information has been tightly tied to two things: the semantics derived from, and the syntax of the exiting technology used to describe, the business processes. This situation is untenable as businesses need to leverage new technologies as they arise without having to re-engineer business processes.

Often different applications use different representations of the same data and one of the hardest and most time-consuming problems for systems integrators is to map one representation to another. An NGOSS system contains a Data Steward (or Data Service Provider) framework component which holds the master copy of the data. Meta-data about the data (descriptions of the data) is stored in the Registry. This also contains the locations/addresses of specific Data Steward(s) for each kind of data.

The Distributed Management Task Force (DMTF) defines standards for the unified management of applications, computers and networks, primarily aimed at the enterprise management market. It has defined an extensible Common Information Model (CIM) [4] which underlies the TMF's common data model.

### 4.2 Migration Issues

All components to date are bound to be 'legacy' as the NGOSS Architecture is not stable. It is imperative that an NGOSS is able to integrate with legacy applications and provides a migration route. This is why the commercial off-the-shelf components that are used require an adapter which is essentially an Application Programming Interface (API) but with a difference. Realistically where the common information model is not used there is a need to map between it and a proprietary data model, which is done in the NGOSS adaptor.

## 5 Web Services

NGOSS can not be treated in isolation from wider e-commerce initiatives. Universal Discovery, Description and Integration (UDDI) [5] is an effort to enable companies to discover trading partners through distributed repositories like Domain Name Servers. Information in these repositories will describe the business processes, service descriptions and binding information needed to do e-commerce with that company. There are many similarities between UDDI and NGOSS.

XML is one technology that can be used to represent NGOSS contracts. As it is self-descriptive it is very powerful but also exceedingly flexible. This flexibility needs to be appropriately bounded so that distributed systems can understand unambiguously what is going on. Provided the XML schemas used in an NGOSS are closely aligned with the agreed common information model, the vocabulary will be understood. But several aspects still need to be considered. They are naming conventions, grammar and the point at which XML documents are parsed.

## 6 Conclusions

There are two main areas of NGOSS that require closer attention; the issues of global state and how business level processes will be translated into systems level processes and relationships between components. Policies can trigger branching of processes but it is not yet clear how business-wide policies, particularly relating to security, will be enforced on processes which at a high level may not seem to be relevant.

## Acknowledgments

My thanks go to the TMF for waiving the cost of membership while I finish my doctorate. Several staff members who have been invited to speak at pertinent conferences have extended me their guest invitations for free for which I am also very grateful. Without the continuing support of my colleagues in the TMF I would not be able to keep abreast of the real and pressing concerns of the industry to the extent that I have.

## References

[1] New Generation Operational Support Systems, GB920 - http://www.tmforum.org

[2] UN/CEFACT Modelling Methodology, N090 - http://www.ebxml.org

[3] System Integration Map, GB914 - http://www.tmforum.org

[4] CIM Specification, DSP0004 - http://www.dmtf.org

[5] UDDI v3.0 - http://www.uddi.org/