Next generation
OSS/BSS architecture

November 25, 2013
Next generation OSS/BSS architecture

Breaking down the silos of operations and business support systems (OSS/BSS) to form an integrated, cross-functional platform that can take a product from conception to execution in a simplified and consistent manner will cut time to market from months and years to weeks.

The systems that keep networks running and profitable are in the direct line of fire when it comes to implementing change. So, as the world moves toward global connectivity, as smartphones cause a shift in user behavior, and as subscribers demand more personalized products and even greater control, the functions of OSS/BSS – such as planning, configuration, fulfillment, charging, billing and analytics – need to be integrated.

A consolidated architecture is a typical computer-science approach for bringing together the functions of different systems. By adopting such a consolidated architecture for OSS/BSS, operators will be able to maintain control over costs while implementing network changes effectively.

The challenges of evolution

By exposing the functionality and information held in their networks, operators have the opportunity to create innovative and ever more complex value chains that include developers, OTT players and subscribers. In these new value chains, the flow of information and control shifts from unidirectional to multidirectional, and participants can be consumers of services and information as well as being producers of them.

New business models for network evolution are based on providing anything as a service (XaaS) – including IaaS, PaaS, SaaS and NaaS – and when using this model, it is not just value chains that become more complex; the life cycles of products and services also become more diversified.

How then, as business models advance, should OSS/BSS requirements evolve to cater for factors such as big data, personalization and virtualization? The simple answer is through configurability. To create a high level of flexibility, the evolution of OSS/BSS needs to be configuration driven, with an architecture based on components.

The impact of big data

Information is a critical resource. Good information is a key asset – one that can be traded, and one that is critical for optimizing operations. As volumes rise, the rate of creation increases, and a wider variety of data that is both structured and unstructured floods into OSS/BSS, access to storage needs to be effortless. In this way, tasks and optimization processes can maximize the use of existing infrastructure and keep data duplication to a minimum.

Data management needs to be secure and controllable, ensuring that the

**Box A** Terms and abbreviations

<table>
<thead>
<tr>
<th>BO</th>
<th>BPMN</th>
<th>BSS</th>
<th>CEP</th>
<th>CLI</th>
<th>(E)SP</th>
<th>ETL</th>
<th>eTOM</th>
<th>GUI</th>
<th>IA</th>
<th>IaaS</th>
<th>IM</th>
<th>JEE</th>
</tr>
</thead>
<tbody>
<tr>
<td>business object</td>
<td>Business Process Model and Notation</td>
<td>business support systems</td>
<td>complex event processing</td>
<td>command-line interface</td>
<td>(enterprise) service bus</td>
<td>extract, transform, load</td>
<td>enhanced Telecom Operations Map</td>
<td>graphical user interface</td>
<td>information architecture</td>
<td>infrastructure as a service</td>
<td>information model</td>
<td>Java Enterprise Edition</td>
</tr>
<tr>
<td>LC</td>
<td>LDAP</td>
<td>M2M</td>
<td>NaaS</td>
<td>NFV</td>
<td>OLAP</td>
<td>OLTP</td>
<td>OS</td>
<td>OSGi</td>
<td>OSS</td>
<td>OTT</td>
<td>PaaS</td>
<td>PO</td>
</tr>
<tr>
<td>life cycle</td>
<td>Lightweight Directory Access Protocol</td>
<td>machine-to-machine</td>
<td>network as a service</td>
<td>network functions virtualization</td>
<td>online analytical processing</td>
<td>online transaction processing</td>
<td>operating system</td>
<td>OSGi Alliance (formerly Open Services Gateway Initiative)</td>
<td>operations support systems</td>
<td>over-the-top</td>
<td>platform as a service</td>
<td>purchase order</td>
</tr>
<tr>
<td>RAM</td>
<td>SaaS</td>
<td>SBVR</td>
<td>SDN</td>
<td>SID</td>
<td>SLA</td>
<td>SQL</td>
<td>TCO</td>
<td>TMF</td>
<td>UI</td>
<td>VM</td>
<td>VM</td>
<td>XaaS</td>
</tr>
<tr>
<td>random access memory</td>
<td>software as a service</td>
<td>Semantics of Business Vocabulary and Business Rules</td>
<td>software-defined networking</td>
<td>shared information/data model</td>
<td>Service Level Agreement</td>
<td>Structured Query Language</td>
<td>total cost of ownership</td>
<td>TeleManagement Forum</td>
<td>user interface</td>
<td>virtual machine</td>
<td>anything as a service</td>
<td></td>
</tr>
</tbody>
</table>
systems accessing information do not jeopardize data integrity and subscribers can feel confident that their information is protected.

The impact of subscriber needs
Personalized services and superior user experience are key capabilities for business success and building loyalty. Subscribers want to be in control, and feel that their operator provides them with reasonably priced services that meet their individual needs, over a network that delivers near real-time response times. The ability to create and test services in a flexible way with short time to market will help operators meet changing user demands.

The impact of M2M
As the number of connected devices gets closer to 50 billion, the need for automated and autonomous behavior in processes such as configuration and provisioning is becoming more significant. Being able to remotely configure, provision and update millions of devices without impacting the network supports scaling while maintaining control over opex.

The impact of virtualization
As a result of virtualization, operators, partners and even subscribers (in the future) can create instances of their services and networks on demand. So, as networks continue to move into the cloud, and SDN and NFV technologies become more widespread, the number of entities managed by OSS/BSS will rise by several orders of magnitude.

So, to help operators remain competitive, next generation OSS/BSS need to fully address the challenges created by certain aspects of network evolution, including virtualization, big data, M2M and personalization.

Making good use of technology
One way to address these challenges is to make good use of advancing technology, particularly when it comes to OSS/BSS implementation architecture. And it’s not just about using technology development in a smart way; it’s also about understanding the potential of a given technology. So, when a new concept results in a significant breakthrough, the services and products that can be created as a result should be readily definable.

Capitalizing on increased flexibility and agility made possible by new technologies (such as virtualization and SDN) needs to be coordinated through a management function, which puts new demands on OSS/BSS architecture.

The evolution of virtualization
The demands created by increasing virtualization of data centers, not just in terms of computational capacity, but also in terms of storage and networking capabilities, are:

- virtualization of the OSS/BSS, and running these systems in the cloud:

![FIGURE 1 Separating planes in SDN architecture](image1.png)

![FIGURE 2 Abstraction of a typical OSS/BSS deployment](image2.png)
tration of resource allocation.

The merger of two giants

management of cloud-based OSS applications such as service assurance; and

management of cloud-based BSS applications such as IaaS and PaaS.

It may, however, not always be beneficial to run certain network elements on generic IaaS resources. For example, information stored in a database may be better provided in the form of a service to subscribers in an IaaS environment, rather than as virtually deployed tenants. The general rule is that anything provided as a service, which is implemented by a piece of software running in a generic IaaS environment, has a reduced level of control and efficiency. Due to the extra layers created by running software in a generic environment, the drawbacks of this approach must be weighed carefully against the benefits of increased flexibility and better (shared) use of physical resources.

For next generation OSS/BSS, the focus should be placed on implementing flexibility in an efficient way together with automation and orchestration of resource allocation.

The hypervisor approach to virtualization, where virtual machines (VMs) share the resources of a single hardware host, is evolving so network infrastructure is becoming more efficient. For example, the failover capabilities of the hypervisor can place agents on the host in a similar way to traditional failover clusters, and can monitor not only VM health, but application and OS health as well. Such features are prerequisites of an efficient virtual environment. However, application architecture may have to take these features into account, as in some cases they cause the responsibility to perform certain tasks (such as data recovery) to shift between the application and the infrastructure.

Service provider SDN

SDN separates the data plane (the forwarding plane or infrastructure layer) from the control plane, which in turn, is separated from the business application plane. As shown in Figure 1, various business applications communicate with SDN controllers, providing a virtualized – possibly hierarchical – view of the underlying data plane.

Generally speaking, the management requirements for SDN and non-SDN architectures are similar, if not the same. For example, both require inventory, ordering and fault management. However, SDN presents a new set of technical issues related to resource management, which brings into question the current partitioning and structure of OSS/BSS architectures.

Specifically, SDN can result in horizontally abstracted virtualized software layers that have limited vertical vision through the hierarchy from the business applications to network devices. So, at the same time as the abstraction offered by SDN makes it easier to expose the capabilities of the network, it creates additional challenges for the OSS/BSS architecture.

OSS always needs to have the capability to map the virtual view of the network to the underlying implementation. Sometimes, the SDN controller handles domain-specific OSS/BSS functionality by hiding parts of the complexity of the control and data planes; and as a result, only a subset of information will be propagated to the business application plane. Sometimes, the underlying layers are not even visible – such as when a third party owns them. In these cases, SLAs can be used to map the virtual view to the underlying implementation.

The evolution to SDN architecture and virtualization causes the number of entities managed by OSS/BSS components to rise, which in turn impacts the way they are managed. For example, a more extensive history of each entity is required, because the semi-static environment used to locate a device using its IP address no longer exists. With SDN, network topology becomes totally dynamic. History data is essential for analysis and management of the network, as this information puts network events into context.

Hybrid flexibility

Modern database design is evolving toward the use of hybrid architectures. This approach allows a wider range of solutions and applications to be created with a single consistent implementation and one logical data store.

Hybrid disk/in-memory databases use in-memory technologies to achieve the performance and low latency levels of an in-memory solution, while still
using disk for data persistency. The hybrid approach allows more data to be stored on disk than can fit into memory; as such, the disk is not a mirror of the in-memory content. This approach is similar to caching disk content, while providing the performance that comes from a true in-memory design – which cannot be achieved by caching disk content alone.

Hybrid SQL/NoSQL (sometimes referred to as NewSQL) solutions are SQL-capable databases that are built using a NoSQL implementation to attain the scalability and distribution that these architectures afford, while still providing support for SQL. However, such hybrid solutions can be limited by their lack of support for partition-wise joins and subsequent lack of support for ad hoc queries – although there are exceptions to this. Hybrid OLTP/OLAP solutions aim to merge the typical characteristics of transactional OLTP workloads and OLAP-based workloads (related to analytics) into a single implementation. To build such a structure typically requires that both database architectures be considered from the outset. Even if such solutions exist, it is difficult to build this type of hybrid from the starting point of an OLTP- or OLAP-optimized architecture.

**Big data**

Modern data centers are designed so that increasingly large amounts of memory with low levels of latency are being placed ever closer to computational resources. This greatly increases the level of real-time processing that can be achieved as well as the volumes of data that can be processed. Achieving these processing levels is not simply a matter of the speed at which operations can be carried out; it is also about creating new capabilities. The developments being made in big-data processing have a significant impact on how next generation OSS/BSS architecture can be designed.

Fast data – the velocity attribute of big data – is the ability to make real-time decisions from large amounts of data (stored or not) with low latency and fast processing capabilities. Fast data supports the creation of filtering and correlation policies that are based on – and can also be adjusted to – near real-time input.

Another big-data concept combines the in-memory/disk hybrid with the OLTP/OLAP (row/column) hybrid to achieve a single solution that can address both OLTP and demanding analytics workloads. Coupled with the huge amounts of memory that modern servers can provide, this approach removes the need for a separate analytics database.

**The business logic**

When OSS/BSS are deployed, they bring business and technical stakeholders together and allow them to focus on the design and implementation of their unique business. The functionality provided by OSS/BSS must support the necessary user-friendly tools to implement and develop business logic. The deeper and more flexible this support is, the more business opportunities can be explored, and the more profitable an enterprise can be.

**Figure 2** shows an abstraction of a typical OSS/BSS deployment. Current implementations tend to be multivendor; with multiple systems performing similar tasks. Organic growth has led to a lack of coordination and as a result, significant time and effort is spent on integration, time to market is slow, and TCO tends to be high. Quite often there is a significant gap between daily business and the systems used to support it.

To transform such a complex architecture into a more business-agile system requires some evolution. As consolidation is fairly straightforward, this tends to be the first step. However, to succeed, it requires systems to be modular, to be able to share data, to use data from other sources, and to have a specific role. Another approach is...
approach to next generation OSS/BSS is to extract hard-coded business logic from the underlying systems, and to structure functionality according to design and life-cycle flow. To achieve this and build an abstract and virtual view of the business successfully, a common, shared and semantically rich information model (IM) and a defined set of relationships is essential.

The information model is the fundamental component of Ericsson's approach to next generation OSS/BSS.

**Building blocks**

When the hard-coded business logic is extracted the following building blocks are created:

- **actors and roles** – such as companies, functions, individuals and customers, suppliers and service providers;
- **services and functions** – such as sales and contracting;
- **processes** – such as TMF eTOM;
- **business objects** – such as products, orders, contracts and accounts; and
- **rules** – such as pricing, prioritization and product/service termination.

These building blocks can then be used to form processes, as Figure 4 illustrates. To achieve the full degree of flexibility, the building blocks for the complete life cycle (from definition to termination) are needed.

**Conception to execution**

Figure 5 illustrates design chain functionality. This process is relied upon to take a new business idea from conception to execution.

Business logic is defined, designed and implemented in what Ericsson refers to as the enterprise business studio. The studio comprises a set of integrated workbenches that have access to all building blocks and information elements, and provide feedback to the process owner as business logic is implemented.

In the Ericsson model, business logic (such as verification, commissioning and decommissioning, supervision, migration and optimization) is transferred to a management function for implementation and application. The management function is also responsible for transferring business logic to the proper execution engines. For example, pricing rules are used in execution by the rating engine, contracting engine and the sales portal engine. Given the potentially massive spread of pricing rules, getting it right at this stage of development is key.
Information architecture

Generally speaking, an enterprise defines the information it needs to operate, and the OSS/BSS manage this information, based on a range of business models. The spread of information across any given enterprise is extensive, and can span many different functional areas from marketing, ordering, strategy and HR, to production and finance. Information models used in OSS/BSS include:

- the enterprise vocabulary and concepts at the business level – for example, an enterprise might refer to a voice product using its marketing name, such as Family and Friends;
- the canonical concepts at the application level – which might refer to the Family and Friends product as family-group; and
- the multivendor concepts at the application level – where the concept of Family and Friends has different names at the business level and the canonical level.

As operators continue to differentiate and offer ever more complex products and services, the requirements on information change. Information is no longer just mission critical; it is also enterprise critical, and changes constantly as business needs evolve.

The shift to next-generation OSS/BSS changes the way enterprise-critical information needs to be handled, creating a number of system requirements:

- information and applications need to be separated;
- the entire life cycle – from definition to termination – needs to be modeled;
- information needs to be shared among all enterprise users;
- master data needs to be determined, and even multiple masters need to be supported to align with different enterprise functions; and
- information needs to be characterized in terms of size, throughput, quality, reliability and redundancy across the board, for all functions and applications – one instance that can be used by all.

As Figure 6 shows, information held in an enterprise can be generated by many sources – both internal and external – and can be classified according to its properties and type. For example, information can be static, structured, event-driven or transactional.

The best way to meet the new requirements on information – driven by the need to differentiate – is to design OSS/BSS in a way that is independent of functional applications with a centrally managed information architecture (IA) that has a common and shared information model.

The key characteristics of this architecture are:

- integrated information across functions;
- information offered as a service – facilitating high-level abstraction and avoiding the need to understand low-level data constructions; and
- information published in catalogs – formalizing IaaS and enabling use by process owners.

Due to the complexity and widespread nature of information models, a modularized IA is needed – one that can be configured to meet the varying needs of enterprises and used in multivendor scenarios with varying information life cycles.

Modular layers

A modular IA can be implemented by categorizing information into a matrix. The first step is to categorize information into (horizontal) layers, where each layer is populated by a number of entities. Subsequently, these entities can be combined (vertically) into a solution.

As Figure 7 shows, the data vault is placed at the bottom of the architecture hierarchy. The most efficient type of storage can be chosen from a number of components, including disk, RAM or virtual resources.

A level up from the data layer is the engine layer, which comprises a set of components that provide access to the information storage. SQL, LDAP, NoSQL or HBase are examples of technologies used in this layer, and each group of data selects the technology that best matches the access requirements for that data.

The grid layer, above the engine, is where the information model becomes accessible. Here, the IM is divided into a set of responsibility areas, such as...
as enterprise catalog with product, service and resource specifications or inventory data. These areas deploy components that expose information as a service and protect the underlying data, ensuring that it is consistent and available for all authorized applications.

The applications that consume and produce information sit on the top of the architecture. They set the requirements on the grid layer so that information is made available with the right characteristics and accessibility according to the needs of the given application. Applications can of course use local caching, but the data vault allows information to persist and be made available throughout the system.

The right-hand side of Figure 7 shows the set of functions that support the flow of information in and out of the model. Typically, this part of the OSS/BSS interfaces to data owned by legacy or other systems, and allows information to be accessed from the outside. Grid components are responsible for interacting with external data sources, and exposing access to information residing in the model. Typical functions include transformation, protocol adaptations, handling of services or data streams and identity mapping, which are also used by applications in the application layer.

Management functions such as definition, registration, discovery, usage, archiving and decommissioning are shown on the left of the IA. As information is no longer hard-coded in each application, but shared among applications at all stages of the life cycle, the management function is vital for ensuring data consistency.

**Deployment stack**

To serve next generation OSS/BSS requires a state-of-the-art deployment stack. A functional view of such a stack is illustrated in Figure 8. The deployment stack should provide a consistent user experience for all processes – from business configuration and system provisioning, to operations, administration and management. It should support applications deployed on a variety of different infrastructures including cloud, and virtualized and bare metal hardware. The deployment stack should provide a means of efficient integration among applications, and enable service exposure in a uniform way.

By making use of the common services provided by a deployment stack that supports scale-out architecture and meets the needs of big data, application development should become more efficient. The stack should integrate easily with existing enterprise systems – a capability that becomes more significant as OSS/BSS are developed and used in multiple scenarios around the world and deployed in an IaaS manner.

**Hardware**

Using existing hardware infrastructure for OSS/BSS deployment is the best option for operators as it consolidates the use of hardware, and supports rapid reconfigurability and scalability. Linux is an attractive OS, as it is community driven and supports the decoupling of software and hardware elements.

**Middleware**

A common and well-composed middleware provides: a consistent environment, effective management, ease of integration, greater availability, better scaling, load balancing, simplified installation, upgrade and deployment, and improved backup capabilities. Such an environment can be provided by either an OSGi container or a JEE application server pre-integrated with availability management, software management and backup/restore capabilities.

**Operations and management**

This function provides common management services for configuration, logging, configuration management, and fault and performance management.

**Application services**

This layer provides common functions for OSS/BSS applications, such as:
- service performance – which monitors and reports uptime;
- licensing – which enables provisioning,
monitoring, control and reporting of licenses;
> user management – which provides authentication and authorization; and
> coordination service – which provides inter-application coordination in a
distributed environment that supports changing license requirements created
by business models such as pay-as-you-grow.

Presentation layer
The presentation layer provides support for GUI, CLI and M2M interfaces
for OSS/BSS applications. A common GUI framework, together with single
sign-on for the entire stack, is key to providing a consistent user experience.

By exposing interfaces to other applications in a uniform way, the amount
of application-application integration required is reduced significantly.

Proposed architecture
The architecture of next generation OSS/BSS is illustrated in Figure 9. At
the business level, the proposal supports service agility in all processes
from conception to retirement and all relevant phases – including planning,
deployment, customer on-boarding and assurance.

The proposed architecture comprises a set of application functions, which
implement tasks such as enterprise catalog, charging, billing, order manage-
ment, experience and assurance. Application functions are implement-
ed through a set of components that can be configured and assembled so they form a complete solution that can also be integrated with existing sys-
tems. The set of common application functions, including correlation and
event handling, support the specific OSS/BSS application functions.

The information architecture separates the information model so information is matched to the application functions, supporting modularity and enabling integration in the overall information model. The information model is based on a shared informa-
tion/data model (SID), with extensions embracing more standard industry
information models.

To support effective implementa-
tion, all components should be
prepared for cloud deployment.

References
Jan Friman

is an expert in the area of user and service management at Business Unit Support Systems (BUSS). He has held various positions in the area of OSS/BSS at Ericsson for 16 years, including R&D, system management and strategic product management. He is chief architect for information architecture at BUSS and holds an M.Sc. in computer science from Linköping Institute of Technology, Sweden.

Munish Agarwal

is a senior specialist in multimedia architecture and chief implementation architect for OSS/BSS. He has been at Ericsson since 2004, working in the OSS/BSS area. He is currently driving the BOAT implementation architecture and is the product owner for Next Generation Execution Environment. He holds a B.Tech. in material science from the Indian Institute of Technology, Kharagpur, India.

Edvard Drake

is an expert in the area of hardware and software platform technologies, and is chief architect for implementation architecture at BUSS. He has 20 years’ experience at Ericsson, ranging from AXE-10 exchanges to today’s commercial and open source technology innovation. He holds a B.Sc. in software engineering from Umeå University, Sweden.

Lars Angelin

is an expert in the technology area multimedia management at BUSS. Lars has more than 28 years of work experience in the areas of concept development, architecture and strategies within the telco and education industries. Lars joined Ericsson in 1996 as a research engineer, and in 2003 he moved to a position as concept developer for telco-near applications, initiating and driving activities, most of them related to M2M or the OSS/BSS area. He holds an M.Sc. in engineering physics and a Tech. Licentiate in tele-traffic theory from Lund Institute of Technology, Sweden.