Event Driven Architecture Augmenting Service Oriented Architectures

Badri Sriraman
Lead IT Architect
(Unisys)

Rakesh Radhakrishnan
Enterprise IT Architect
(Sun Microsystems)

January 2005

Table of Contents

Introduction and Overview

This paper attempts to highlight the significance and value proposition of leveraging the fundamental principle's and concepts behind "Event Driven Architectures" (EDA) for "Service Driven Architectures" – what is more commonly referred to as "Service Oriented Architectures" (SOA), for XML based Services. "Web Services" being one of the later instantiations prior to which we've had CORBA or JINI based Services Oriented Architectures. This paper highlights the fact that EDA augments SOA – and MDA (Model Driven Architectures), EDA, SOA –all approaches compliment and support User Centric Agile Solutions.

In ***Service Oriented Architectures (SOA)***, an Enterprise's Architecture is developed in a Service Driven approach, where each service is autonomous and can be considered to be a Service Building Block (SBB). Using an analogy between the concept of service and a business process, in SOA, loosely coupled SBB are orchestrated into business processes that support customer goals and/or organization's business goals. More than just a component of reusable code, a SBB becomes part of a running program that can be invoked by a client without having to incorporate the code itself. A SBB by definition is reusable and replaceable, that is, one SBB's service is reused again and again by other services for the functionality it provides and SBB's service (provider implementation) can be replaced by another (another providers implementation). In SOA – the SBB's themselves can be categorized into basic/foundation services, management services, security services, business services, portal services, etc. It should also be noted that a SBB is offering a specific functionality for an Enterprise and transcends projects, i.e., a Digital Rights Management Service (DRM) as a SBB is only implemented once in an enterprise architecture and can be reused across projects that deal with delivering content, services, multi-media, etc. In SOA the communication flow is closed to new unforeseen inputs once the communication flow has started, i.e., they are typically well defined and the boundaries well set.

Event Driven Architecture, embraces mechanisms for coordinating the callers and providers of service, producers and consumers of data, sensors and responders of software events with variable level of communication coupling, with variable spectrum of message correlation and with variable options to deliver quality of service. EDA engenders a network that listens to thousands of incoming events from different sources, wherein complex event processing results in intended system response. EDA supports dynamic, parallel, asynchronous flows of messages and hence reacts to external inputs that can be unpredictable in nature. For example EDA approach can enable a caller invoke a provider SBB using events without knowing who provides it or what address the provider uses, what choice amongst multiple providers exists, while load balancing across them and selecting amongst these providers with varying levels of qualities of service based on the caller's requirements. Meanwhile the same software event that represents the request or the events generated by the service in response can be of interest to other SBB in the network, opening a channel for value-add to the customer and business. An EDA can coordinate synchronously or asynchronously between software endpoints, and possibly provide both synchronous and asynchronous access between the same participants. In EDA, simultaneous streams of execution can run independently of each other to fulfill a customer request or system responsibility, typically an event bus serves as a platform to manage integration and/or choreograph a larger process.

In ***Model Driven Architecture***, modeling an enterprises system (data, systems, and model of your data model) in conjunction with meta-data, which keeps a record of an enterprise's architecture in-terms of data, information, application, services, technology and implementation (platform specifics) is the basis. There are three basic tools used in an MDA as defined by OMG (Object Management Group) – (see http://www.omg.org/mda): ***Meta-Object Facility (MOF)-*** The MOF defines a standard repository for meta-models and, therefore, models (since a meta-model is just a special case of a model)**. XML Meta-Data Interchange (XMI)-** XMI defines an XML-based interchange format for UML meta-models and models; by standardizing XML document formats and DTD (document type definitions). In so doing, it also defines a mapping from UML to XML. **Common Warehouse Meta-model (CWM)-** The CWM standardizes a complete, comprehensive meta-model that enables data mining across database boundaries at an enterprise and goes well beyond. Like a UML profile but in data space instead of application space, it forms the MDA mapping to database schemas.

An example Scenario where Events Augment Services:

Today within Telco environments services such as Identity, Location, Presence, Weather, have been implemented as a re-usable service, i.e., the location logic running in a Location service can be re-used for many other services – such as location based weather report, location based mapping, location based retail list, etc. Now as an end user I travel to a particular city – such as Seattle for an important meeting the next morning. I setup my alarm for 6:00am on my cell phone and along with my alarm; I receive a short weather report and a Doppler image for the specific location that I'm in. Majority of this scenario is made possible due to a Service Oriented Architecture (SOAP/XML). Now by adding the concepts behind Event Driven Architecture, this scenario can get even more interesting. Lets say due to time difference (5:30PST is actually 8:30EST) I actually wake up before the alarm goes off, and access/login to a TV/STB. With my presence in the network, now, I only receive a weather report/Doppler image on my cell phone and not the alarm, since an event gets triggered the minute I login stating that I'm present and hence awake (canceling the wake-up call). This could be taken one step further, when I not only access TV/STB services, but I actually view a weather channel report. Based on my consumption of this service another event takes place where the weather report/Doppler image along with the wake up call, all gets cancelled. (This is user centric!!). At the same time the flight that was scheduled to depart at 2:00pm that afternoon from Seattle is delayed due to storm in my destination city (Washington DC) and an event is triggered to send me a message notifying the same, with an alternate schedule.

This paper is the second in a series, that includes;

- Model Driven Architecture enabling Service Oriented Architectures
- Event Driven Architecture augmenting Service Oriented Architectures
- Component Based Architecture supplementing Service Oriented Architectures
- Utility Compute Architectures supporting Service Oriented Architectures

All four papers describe the power in SOA when other conceptual architecture patterns associated with MDA, EDA, CBA and UCA are blended with SOA. It adds more meaning to the notion of Service On Demand and True Service Mobility – anywhere, anytime, any network, and any device.

Value Proposition of Events (EDA) for Services (SOA):

In setting the stage and provide a validating scale to the discussion, authors first present how EDA completes and compliments the meta-architecture solution space with SOA and MDA. Then presents an architectural viewpoint framework that will validate while highlighting the value-proposition of EDA for SOA with examples.

Firstly, Authors believe MDA, SOA and EDA form an axis of Architecture Strategies that makeup the evolution of any software architecture in the architectural solution space. This belief stems from the fact that three fundamental orthogonal elements of any software are structure, function and data. Authors believe MDA, SOA and EDA are orthogonal concepts that are evolved forms of these fundamental three. MDA addresses structuring through abstractions; while SOA defines functions in chunks and EDA describes the data in context.

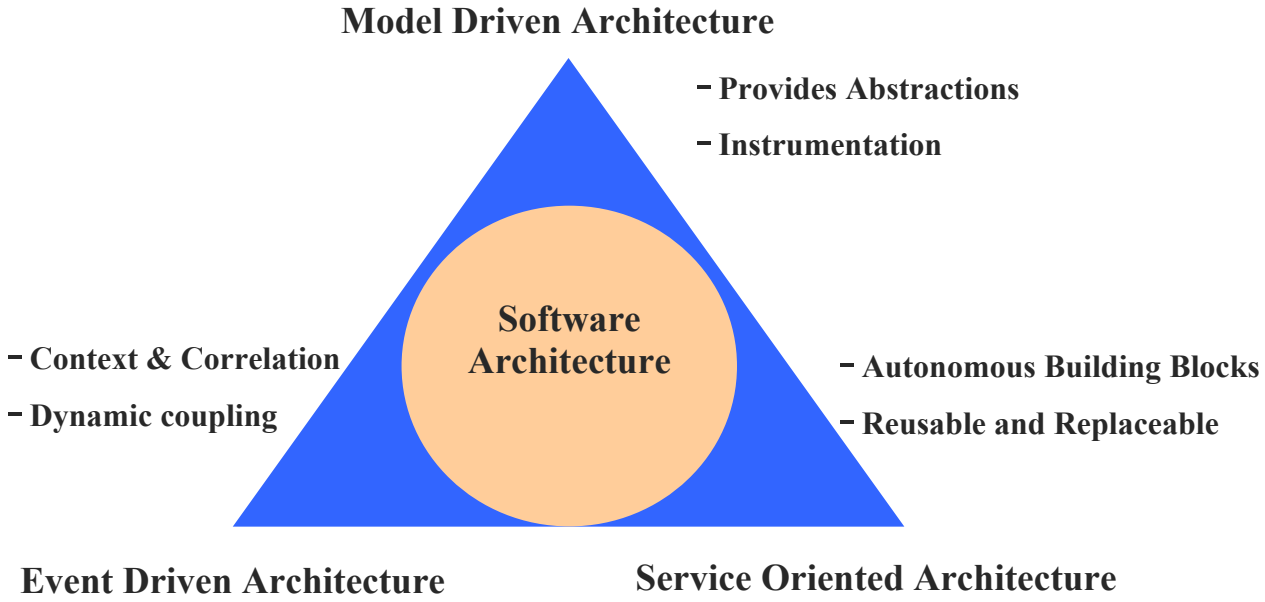

Figure 1: Axis of Architectural Strategies

Secondly, in constructing a validating scale for discussion, authors would like to present similar orthogonal concepts in the architectural concern space. Authors view users, business domain and system builder as three primordial architectural stakeholders whose concerns are usually orthogonal to each other. Users represent the dependency concerns existing external to the system on the system, business domain represent the internal functionality concerns that make up the system and system builder represents the development concerns that exists in evolving the system. Authors believe viewpoints emanating from these architectural stakeholders into the architecture will attain coverage for the discussion, hence validating the supposition that EDA augments SOA.

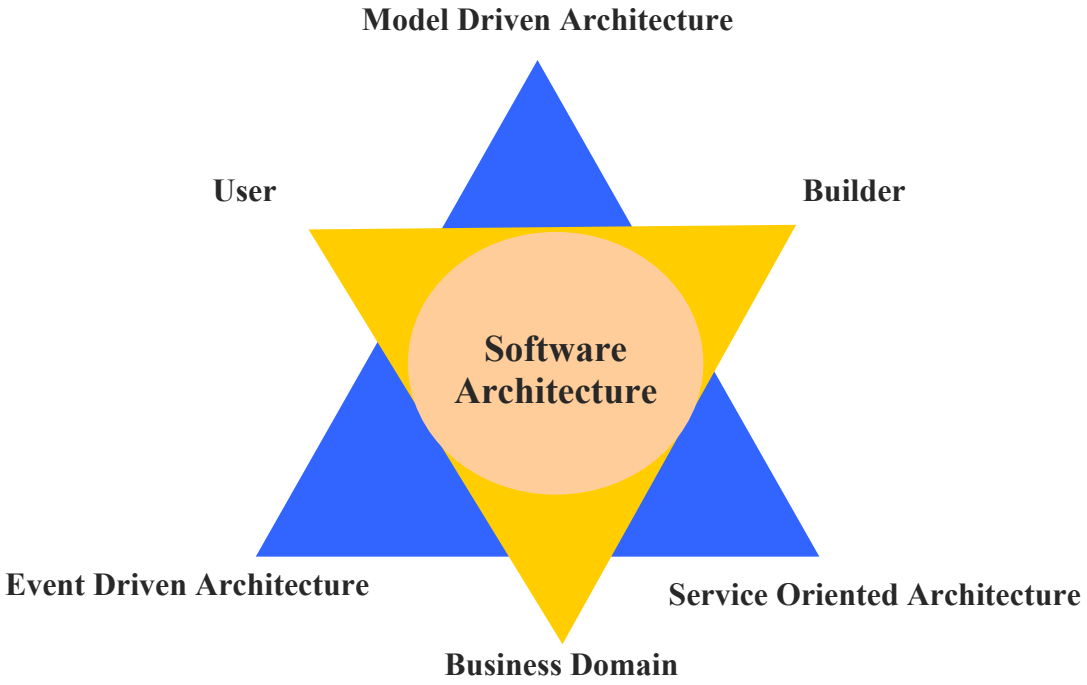

Figure 2: User, Development and Business Domain Perspectives

There are several perspectives from these three architectural stakeholder viewpoints which modeling event based mechanisms add value to services, and they are;

User Viewpoints

- ***User affinity in Services delivery***
  This perspective acknowledges that conversation experience of user spans a wide spectrum; that identity centered service rendering engenders improved quality of service. We explain how events provide opportunity to manage service call life-cycle even across sessions and across devices.

- ***Context-aware responsive service rendering***
  This perspective acknowledges that user service request could be time unbound (like subscribe-notify) and service rendering could span over time; that contextual execution of services is mandatory. We explain how events mechanism transform service access networks into context-aware responsive service rendering networks.

- ***Group Aware Services delivery***
  This perspective acknowledges that users could expect to engage each other in meeting their individual goals; that system handle concurrency on resource it manages while hosting one user as a service to others. We explain how events recognize concurrent service engagement and help discover opportunities for correlation.

Business Domain viewpoints

- ***Coherent Execution and Orchestration of Services***
  This perspective acknowledges that software services at execution time are always part of larger business process satisfying a real-world goal; that multiple system activities spanning over time realize these goals. We explain how virtual event in the system represent real-world events and how correlation of them could help aggregate services to realize business process.

- ***Integration of real assets and virtual Services***
  This perspective acknowledges that software services are allocated and/or bound to real-world elements; that these elements have representation in virtual terms and apply constraints/opportunities on software services. We explain how events provide opportunities to synergize and manage software service on real-world capacities.

System Builder viewpoints

- ***Building agile solutions with dynamic composition of reusable services***
  This perspective acknowledges that SOA builders should focus on modeling SBB to make its internals cohesive; and not focus on context of its use or assembly. We explain how SBB coupling spans a wide spectrum, and how modeling events with SBB meta-data embedded in them can at runtime evaluate context of execution and enhance the ability or the means to assemble SBBs to fulfill solution variability.

- ***Building a higher plane of abstraction using lower-grained Services***
  This perspective acknowledges that SOA builders will build SBBs at level of granularity and concreteness that fulfill current need, that need to re-factor implementation and grow more functionality or assemble new higher-level service from existing is expected. We explain how events can transform and translate context data to fulfill changing contracts of SBB and correlate multiple realizations.

The rest of this paper breaks down 5 of these major value proposition- perspectives, associated with the conceptual architectural pattern within EDA and SOA, namely;

- ***User Affinity in Service Delivery***
- ***Context aware and responsive service rendering***
- ***Coherent Orchestration and Execution of Services***
- ***Agile solutions with dynamic composition of service building blocks***
- ***Higher plane of abstraction for lower grained services***

User affinity in Service delivery

This perspective acknowledges that conversation experience of a user spans a wide spectrum; that identity centered service rendering engenders improved quality of service. We explain how events provide opportunity to manage service call life cycle even across sessions and across devices. From an end users perspective an interaction with a Service (web service, communication service, business service, entertainment service, etc.,) takes place over a network and in majority cases involves complex interactions. The conversation experience of user spans a wide spectrum

- simple request-response call to a service
- multiple user interaction that realize a user goal
- multiple user goal realized overtime and session that realize a entire business interaction
- multiple business interaction with user that profiles to analyze user experiences, anticipate service opportunities and manages user service portfolio

For example, accessing and ordering a book from online bookstore, with a single click (a few clicks), involves book reviews, profile sharing, credit validation, payment, shipment profile, shipping, order completion message, customer service call (to ensure satisfaction), cross-selling/up-selling promotions, OLAP reporting, etc. Multiple modular services are executed, identity assured, profiles shared (including shipping address, credit card profile, user profile), preferences recorded, transactional messages being sent and call's being placed. These modular services interact by exposing their interfaces via a metadata-repository and services are modeled (PIM and PSM) in a common warehouse model. The interaction between all these services can be modeled as event responses during the communication flow between services that are loosely coupled.

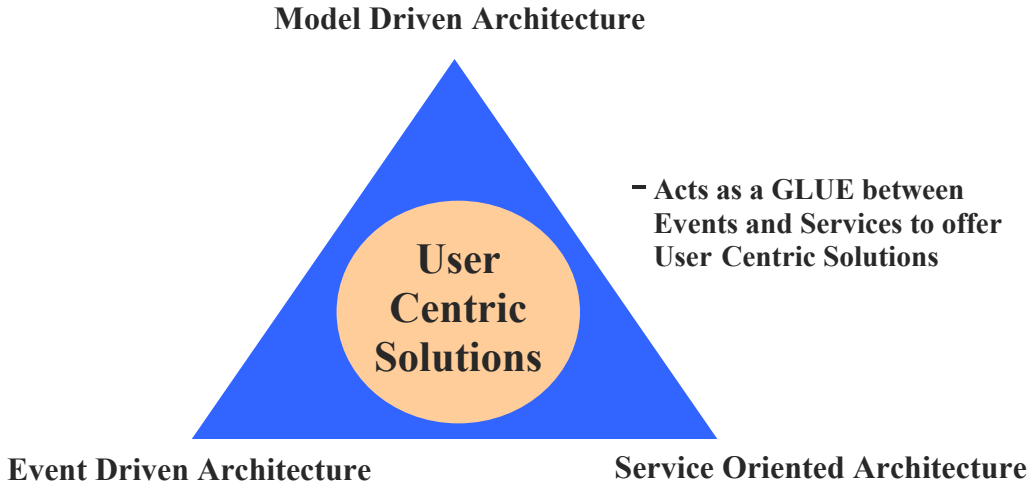

Figure 3: MDA acting as the GLUE between Events and Services to offer Solutions

So we clearly see that MDA acts as the glue between EDA and SOA to offer User Centric solutions, in conjunction with an Identity Enabled Network (IDEN – see paper on IDEN). Events access Services (that are implemented with differing technologies (J2EE, .NET) and differing platforms, via meta-data repositories (that expose the interfaces of these services) and different models (including network models). IDEN is the standards based federated AAA mechanism that supports Single Sign On, session management, profile management, etc. User's preferences, context is taken into account via an IDEN.

Context-aware responsive service rendering

This perspective acknowledges that user service request could be time unbound (like subscribe-notify) and service rendering could span over time; that contextual execution of services is mandatory. We explain how events mechanism transform service access networks into context-aware responsive service rendering networks.

From the overall system perspective delivering services to user requests can involve choreographing a large service calls to prepare a single simple response, immediately or over a period of time repeatedly or a complex response to be delivered over multiple interactions, user sessions and in multiple formats. Such service rendering is possible only when a

- Request-response is shifted to sense-deliver paradigm for services
- Passive Service-networks become proactive service-networks (with the combination of events executed in the core and access networks)

For example, Identity Service Provider (ISP) use IDEN to tracks user time-space, monitor access channels, manage “service delivery” instance life-cycle, analyze customer experiences, anticipate service opportunities and manages user service portfolio. ISP needs to expose user request context providing service networks ability to choreograph intelligent service rendering to enhance user experience. User directed persistent service rendering despite shifting delivery methods, delivery devices, time and location of the subject user is possible only with event enabled service access converting access networks to delivery networks

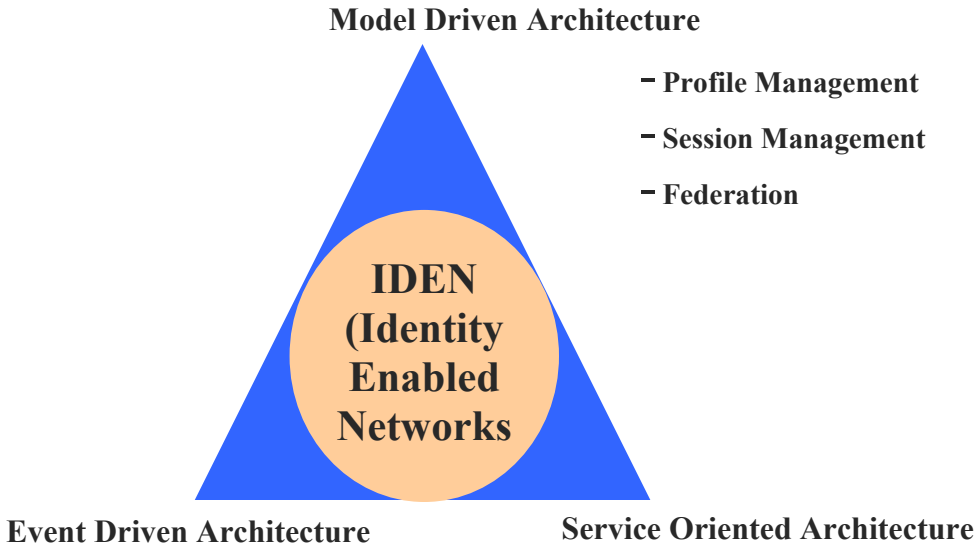

Figure 4: IDEN supports EDA, MDA and SOA

The notion of both Directory Enabled Networks (DEN) and Identity Enabled Networks (IDEN) also come into play here. The Identity/Directory Enabled Network (DEN and IDEN) initiative is designed to provide the building blocks for more intelligent management by mapping concepts from CIM (such as systems, services and policies) to a directory (a highly distributed database), and integrating this information with other elements in the network infrastructure. This includes;

- *Common identity and security administration*
- *Common understanding of managed systems and services*
- *Information related to locations, groupings and policy*
- *Information related to presence*

In conjunction with an Identity System acting as a Session manager that could potentially manages distributed sessions (call sessions, service sessions, user sessions, etc.), contextual session management is made possible. Information pertaining to presence, locations, grouping and more play a significant role in ensuring context prevails within an end-to-end session.

Group Aware Services delivery

A subset within the context aware responsive services is group aware service delivery. This perspective acknowledges that users could expect to engage each other in meeting their individual goals; that system handle concurrency on resource it manages while hosting one user as a service to others. A good example of this scenario is online gaming services that leverage mechanisms from both EDA and SOA. In such scenarios events recognize concurrent service engagement and help discover opportunities for correlation.

Coherent Execution and Orchestration of Services

This perspective acknowledges that software services at execution time are always part of larger business process satisfying a real-world goal; that multiple system activities spanning over time realize these goals. We explain how virtual event in the system represent real-world events and how correlation of them could help aggregate services to realize business process. The idea behind coherent execution and web-service orchestration providing an open, standards based approach for connecting different web services together and to coherently execute higher-level business and communication processes. This is desperately needed in today's world due to the extensive modularity offered by web services in a SOA and the granular functionality that can be provided by each module as opposed to monolithic applications. With EDA a SBB will collaborate and coordinate its activities with other SBB in a composition to achieve a higher-level goal. Well-defined behavioral dependencies and the coordination in time between SBB are of a great importance in achieving the goal. Such event correlations are externalized to a workflow or managed by the SBB container. For, example a simple validation process that occurs before large financial transactions takes place (like a mortgage) can be granularly broken into, address validation service, identity validation service, credit validation service, asset validation service, title validation service, and more, each implemented as web service (with different incarnations in terms of implementations), that is orchestrated and executed as a comprehensive validation business process.

The problem of event correlation spans a wide spectrum

- A simple call level correlation of response and request in asynchronous mode
- A activity level correlation of multiple resource for transaction and compensation
- A business process level correlation of various state transition of a entity life-cycle

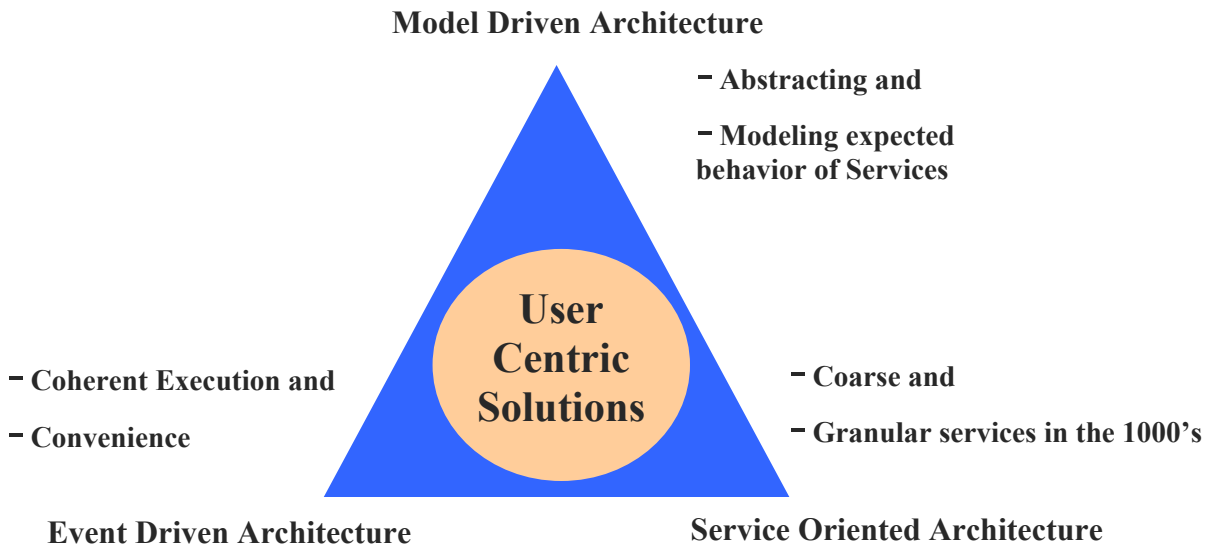

Figure 5: Coherent Orchestration made possible by EDA for SOA

Typically an Enterprise Service Bus (ESB) mechanism, such as (such as JAIN Service Logic Execution Environment), is used for such correlation between callers and service providers in an SOA. It enables a caller to invoke a service without knowing who provides it or what address the provider uses. The ESB can choose amongst multiple providers, load balance across providers and stop using a provider while it is unavailable, and select amongst providers with varying levels of qualities of service based on the caller's requirements. An ESB can coordinate synchronous or asynchronous services, and in fact can provide both synchronous and asynchronous access to the same service. This becomes especially useful when there are several 100 thousand granular services to choose from, when building complex end-to-end business processes that covers the entire spectrum from demand management to the supply-chain.

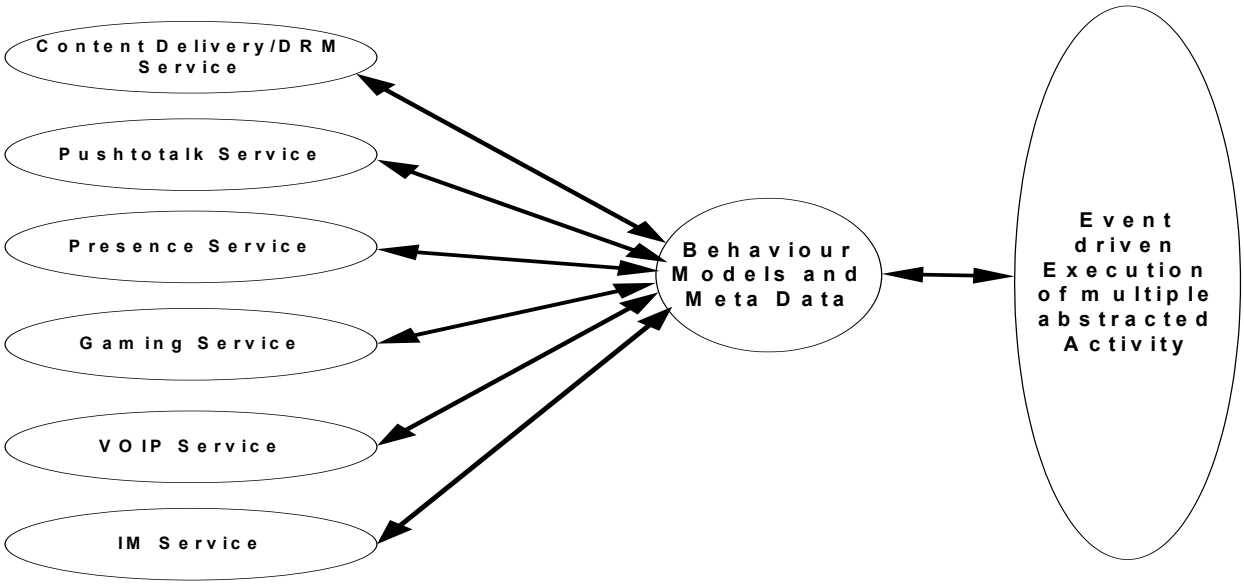

Figure 6: Coherent execution of services (both business and communications)

Integration of real assets and virtual Services

In conjunction with coherent execution and orchestration of services, integration of real-world assets and virtual services can be accomplished. Software integration with real-world assets like devices has always been event-driven. Now plugging SOA approach delivers a value-add that is hitherto unrealized. In SOA with EDA, granular software services are allocated and/or bound to real-world elements; these elements have representation in virtual terms and apply constraints/opportunities that engage software services enhancing the possibilities for extending real-world capacities.

For example, Physical Access Control systems, which are event-driven by nature, can be easily integrated with Logical Access Control of enterprise systems. A user login event to an enterprise system can be correlated to an event generated at the time of user's physical entry to the facility. Infact, failure of this specific correlation can generate an alert event that can be assessed for resource protection rules and vulnerability patterns that will engage proactive response like logical access denial and fraud investigation for security compromise.

This integration could potentially be implemented with Identity Systems that execute policies associated with physical and logical access control. Such as, prior Java SIM Card based authentication to validate access into a physical location as a prerequisite to access specific security sensitive virtual service. This approach of integrating EDA with SOA enhances identity assurance mechanisms, for real world applications.

Building agile solutions with dynamic composition of reusable services

This perspective acknowledges that SOA builders should focus on modeling SBB to make its internals cohesive; and not focus on context of its use or assembly. We explain how SBB coupling spans a wide spectrum, and how modeling events with SBB meta-data embedded in them can at runtime evaluate context of execution and enhance the ability or the means to assemble SBBs to fulfill solution variability. Often in system development we see services built to realize a part of a certain activity or a business process reusable in another context and in the end several contexts. This is the very nature of building solutions based on SOA in an incremental and iterative methodology. Primary issues in inhibiting reuse and evolution of an agile solution are

- SBBs developed for initial context may be of different levels of granularity. If our focus deals only with one SBB and one system we will end up "engineering in the small"
- Coupling models to integrate different SBBs vary widely from simple request/response, publish/subscribe, rule based routing based, reliable messaging and to complex transformation
- Re-factoring is postponed forever because new SBBs could leave broken links hurting overall integrability.

In a EDA augmented SOA, we will externalize the coupling aspects from SBBs and use ESB that use context data embedded within events to correlate and coordinate services, this affords following benefits

- Remove granularity as a hurdle in assembling SBBs
- Provides easy means to dissemble, re-factor and assemble in a model driven approach
- Augment system capabilities in system non-invasively like horizontal or enterprise wide service like Service Management, Service Security, Persistence as one other SBB applied orthogonal as a Aspect wrapping the business services
- Even build system that do opportunistic problem solving, reacting to external inputs that can be unpredictable in nature and use experimentally derived heuristic to control processing to generate responses

The convergence between business environment and communication environments in the form of Service Execution environments for J2EE based business services and JAIN based communication services is achieved first via a common **Identity *System that also holds data and meta-data about users and the services they consume.***

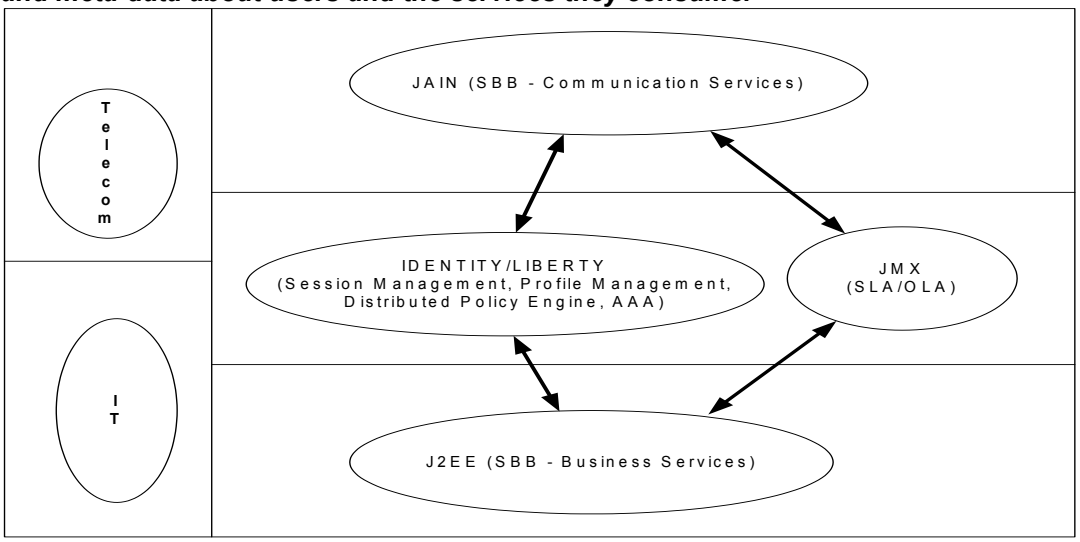

Figure 7: J2EE for SOA and JAIN for EDA leveraging common Identity System

Again modeling the services, its relationships and introducing a level of abstraction bridges the events and the services that are invoked within a Session.

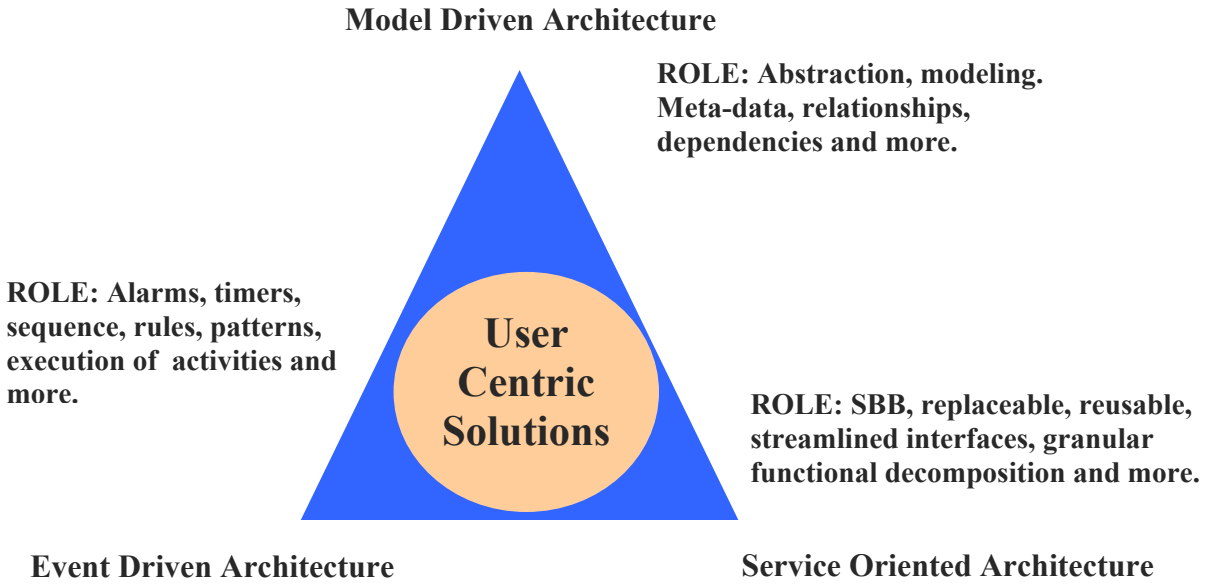

Figure 8: Functional decomposition of Role's between EDA, MDA and SOA

Building a higher plane of abstraction using lower-grained Services

This perspective acknowledges that SOA builders will build SBBs at level of granularity and concreteness that fulfill current need, that need to re-factor implementation and grow more functionality or assemble new higher-level service from existing is expected. We explain how events can transform and translate context data to fulfill changing contracts of SBB and correlate multiple realizations.

SOA architectures support evolving durable contracts to define services making SBBs reusable while separating the implementation that realizes it as replaceable. In that sense contracts serve abstraction to the clients of the SBB from its implementation. In any large systems that grow over a period of time, we will end up with SBBs that have aggregate relationship with other SBBs by its usage in certain context of execution, there by allowing us to treat called SBB, usually lower-grained, as implementation detail for the higher SBB. We can employ EDA approaches to derive behavioral dependency arrived upon by an abstracted SBB based on contextually gleaned from software events and states. Such a system will have its solution-space organized in levels of abstraction implemented by SBBs at different levels.

Below we present a simple an example of using abstracted activity as SBB that coordinate using event mechanism.

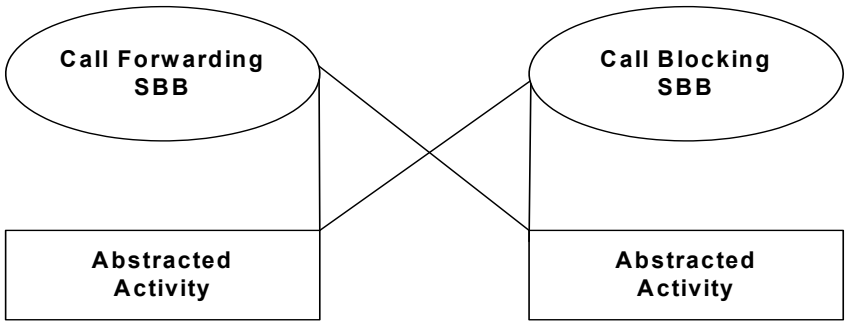

Figure 9: Abstracted execution of related sets of events

A more complex example will be as below.

A high-powered CxO meeting (CEO, CTO, CIO, etc.) in a firm gets cancelled hours before the meeting due to an unforeseen emergency. The Executive administrator cancels this meeting in an event enabled Calendar service. Since these services are both event enabled and identity enabled automatic notification is sent to all attendees in a contextual manner (based on contextual session management) where based on presence and usage of service, appropriate notification is sent. The CEO is playing golf and has his cell phone available – a notification call is sent to his mobile phone, the CTO is in a anonymous IM/chat session with his 300+ employees regarding the new re-org and an IM notification is sent and the CIO is catching the latest CNN news and an appropriate message is sent to his STB/TV.

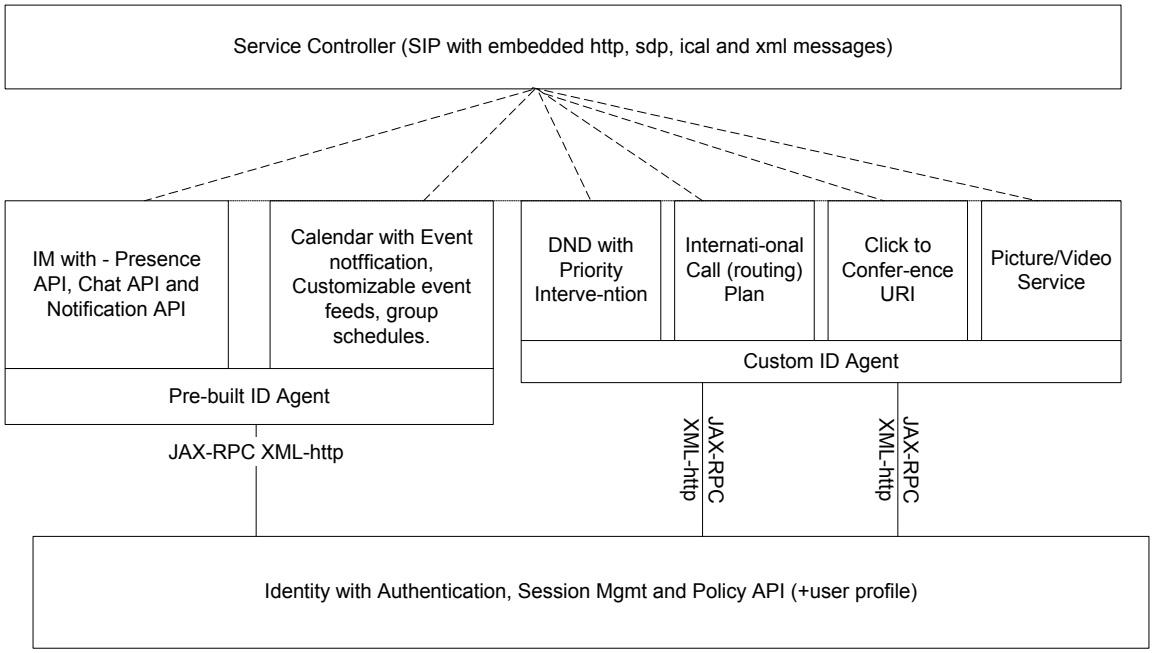

Figure 10: Functional decomposition of Role's between EDA, MDA and SOA

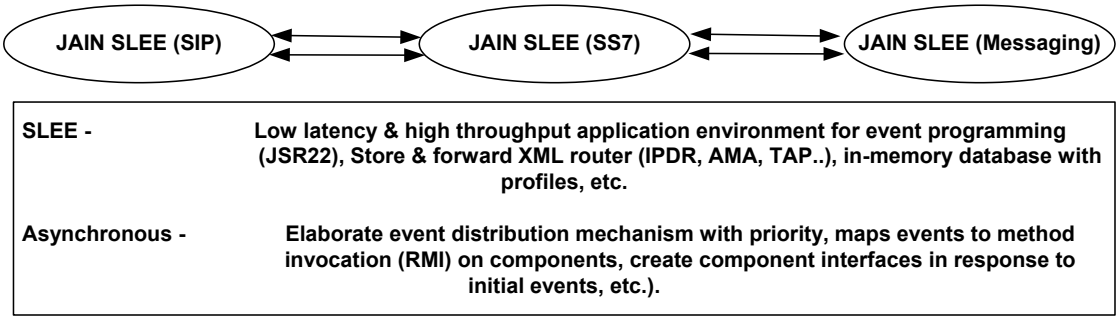

Figure 11: Carrier Grade, low-latency, high throughput, event driven environment

Conclusion

Finally, if Enterprise's embraces all the five areas of synergies discussed in this paper;

Grouping, interlinking and coupling services (brokered SBB) with Service-Metadata/meta-model, addressing User Affinity in Service Delivery, ensuring context aware and responsive service rendering, coherent orchestration and execution of Services leads to agile solutions with dynamic composition of service building blocks that leverage a higher plane of abstraction for lower grained services. Leveraging a DEN/IDEN (directory/identity enabled network) and delivering via an Event Execution environment then the Enterprise (Technology) Architecture is set to achieve Totally Agile Solutions that uses end-to-end integration with straight through processing. All these approaches leads to the strategic benefit associated with EDA augmenting SOA, especially in a web services world, where Services are delivered on demand.

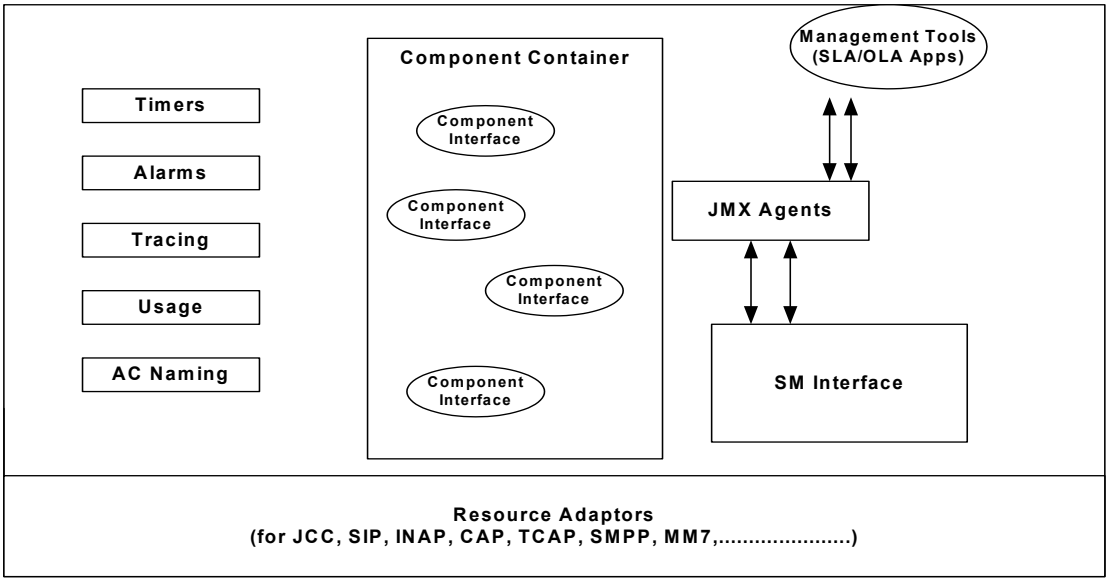

Figure 12: Sample Event Execution Container (JAIN SLEE)

This applies to both communication services and business services and the next generation converged services.

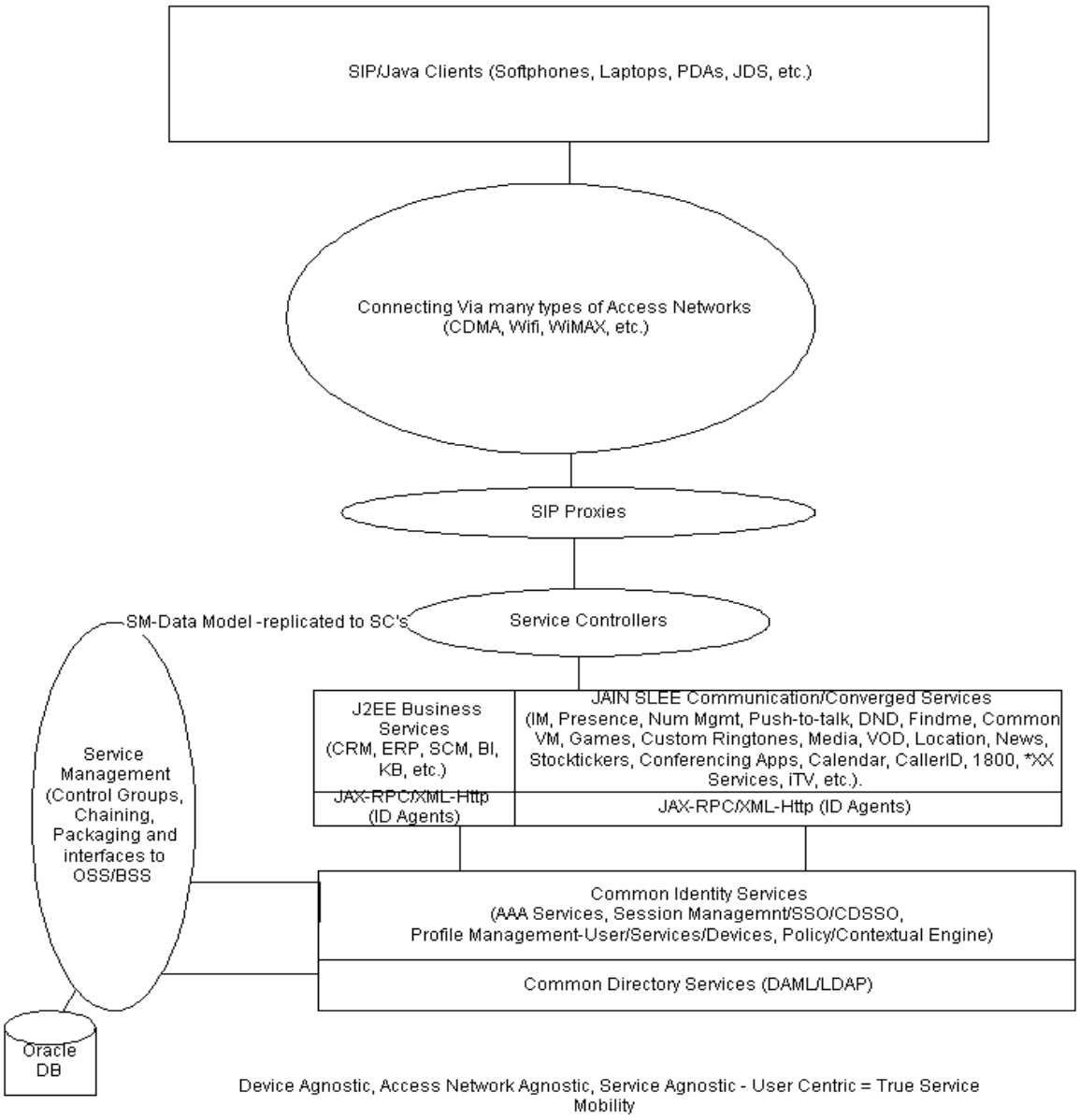

Figure 14: EDA and SOA leveraging a common ID System for SIP Services

End-to-End Java Technologies (the combination making it an ***"Architectural Technology***") ensures that EDA augmentation of SOA's is possible and all the five benefits associated with the same are achievable.

Java's support includes many open standards body, standards forum's and industry initiative's such as OMG's MDA initiative (CWM, MOF, XMI, etc.), TOG's SOA initiative (ADM, TRM, IIRM, etc.), DMTF initiatives around (DEN/IDEN and CIM), ISO initiatives around XML based STP, NIST initiative around JAIN/SIP, JBI initiative (java business integration), and more to deliver (IP/packet) network centric solutions that offer services that are device independent, access network independent, service independent, domain independent and platform independent, however they are profile-driven and context-sensitive to address end users needs. In fact Java has been the inspiration for the initiation and implementation of many of these initiatives. A good example here is J2ME's support for WAP, OCAP and SIP –so a common development environment can exist between Cell phones, STB (set top boxes) and Soft phones. Additional ancillary standards such as, RFID, CDC/CLDC, MIDP, LDAP, XML, SOAP and Http/Http-s are all supported in these Java Technologies.

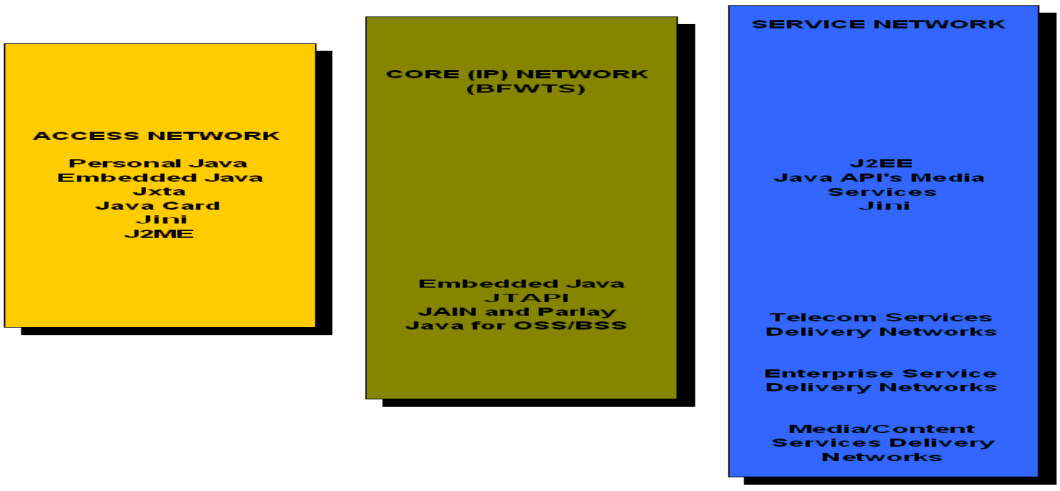

Figure 13: Java Technologies

These Java Technologies work in conjunction with IP initiatives such as MobileIP and IPSec as well as XML technologies including SAML and XKMS.

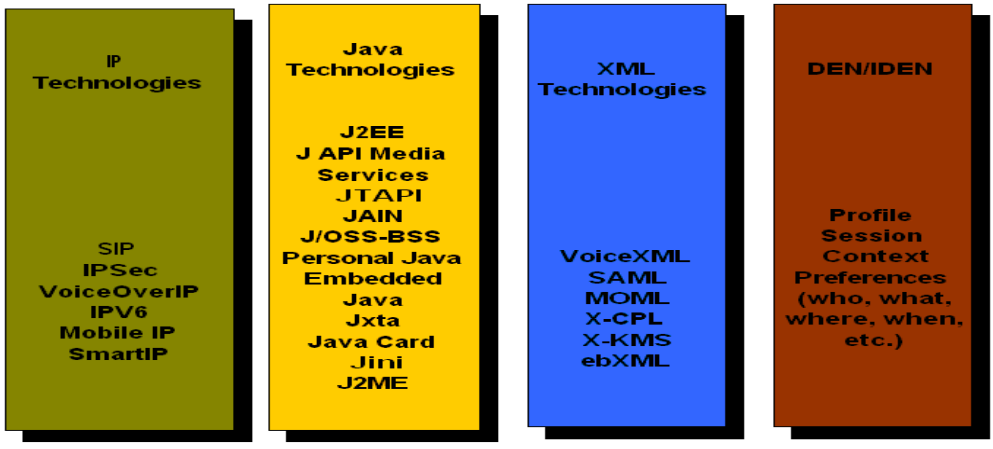

Figure 14: IP, Java, XML and IDEN