WEB PORTAL INTEGRATION ARCHITECTURE APPROACHES

Dr. T. G. K. Vasista[1], Dr. Mohammed A. T. AlSudairi[2]

Vice Rector Office of Business Development,
King Saud University, Saudi Arabia
[1]gtatapudi@ksu.edu.sa
[2]mas@ksu.edu.sa

ABSTRACT

Enterprise Modelling with Web portal integration architecture requires investment of advanced architectural thinking into definition of services before any development of services or service consumers can begin. Service Oriented Architecture (SOA) is gradually replacing monolithic architecture as the premier design principle for new business applications with its inherently systematic nature and capability. Earlier efforts of notable styles of SOA such as CORBA and XATMI have failed to be adopted as main stream projects because of demanding design process requirement with sense-making activities and even have been residing with the modern SOA or Web services middleware. In this paper it is aimed to incorporate sense-making design activities with the proposed semantic web service based architecture. This paper tries to tackle the above problem by proposing a service-oriented architecture for web data and service integration. A gen-Spec architectural pattern has been suggested and adopted in order to tackle the problem. Firstly, it proposes a service-oriented platform independent architecture and Secondly, it presents a specific deployment of such architecture for data and service integration on the web using semantic web services implemented with the WSMO (Web Services Modeling Ontology).

KEYWORDS

Model Driven Architecture, Platform Independent Integration Architecture, Platform Specific Integration Architecture, SOA Architecture, Web Portal Integration Architecture Approaches.

1. INTRODUCTION

An enterprise modelling process that follows object oriented approach is a set of partially ordered steps intended to reach the objective of building a fully integrated, dynamic, object-oriented model of the enterprise. An abstract mechanism is proposed to enable this process [16]. Gen-Spec is a fundamental structural relation between two entities denoting the fact that the specialized entities share common features, states and structural and procedural inks with the generalizing entity [17]. The process is usually generic because it applies to most types of enterprises. Enterprise models are the products developed from the process and these can be used by various stakeholders in an organisation for the purpose of providing them (a) an understanding on the enterprise (b) an integrated information systems design (c) a model that can address change management (d) how a model or a framework can be made reusable [16].

Service-Oriented Architectures are particularly well suited to cope up with the needs of such an ongoing incremental process optimization [6] as represented by Capability Maturity Model Integration (CMMI). CMMI in software engineering and organizational development is a process improvement approach that provides organizations with the essential elements for effective process improvement [15] with the characteristics of the maturity levels as given in Table 1.1

Table 1.1 Characteristics of CMMI Levels [15]

Maturity Level No.	Maturity level Title	Maturity Level Description
Level 1	Initial	Processes unpredictable, poorly controlled and reactive.
Level 2	Managed	Process characterized for projects and is often reactive
Level 3	Defined	Process characterized for the organisation and is proactive
Level 4	Quantitatively Managed	Process Measured and Controlled
Level 5	Optimizing	Focus on Process Improvement

When making architectural decisions, one must carefully analyze the advantages and disadvantages of the level of coupling [2, 3]. Generally speaking, OLTP (online transaction processing) style applications, as they are found throughout large enterprises, do not normally require a high degree of loose coupling, as these applications are tightly coupled by their nature. ERP systems are usually found suitable for such kind of single large monolithic representation and handling [6]. But the recent trend in Globalization is increasing the scale and complexity of the modern enterprises by forcing the enterprise to be represented as a global network consisting of multiple units and functions [12, 13]. So when business processes are highly distributed, the different subprocesses and transactions are generally more independent of each other or more loosely coupled. Loose coupling though increases the flexibility, increases the complexity. The enterprise software architecture is the architect's most important tool at hand to confront the changes to and expansion of functionality that increase system complexity and reduce efficiency [11]. Thus in enterprise software, the architect takes on the roles as an outside influencer and controller. It is his responsibility to oversee individual software projects from the strategic point of view of the overall organisation, as well as from the tactical, goal-oriented viewpoint of the individual project. He has to balance different requirements while attempting to create an enduring order within the enterprise software landscape. By technique of refactoring the current solutions, architects constantly strive to reduce complexity and thereby the agility of the system (See Figure 1.1) [6].

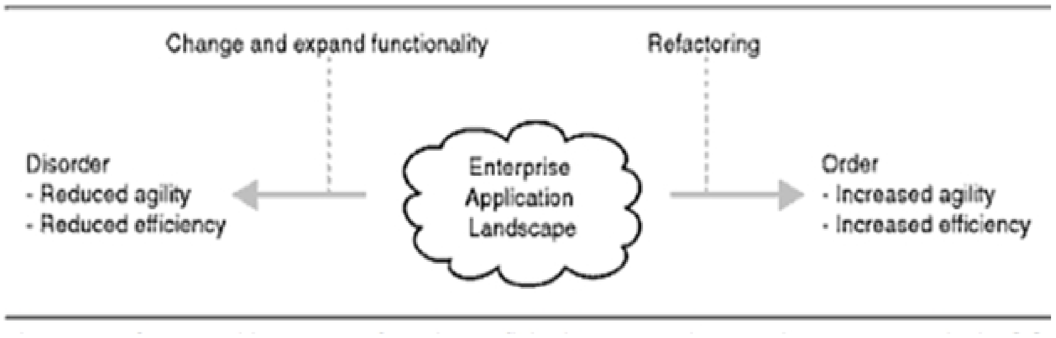


Fig 1.1 Software architects use refactoring to fight the constant increase in system complexity [6]

Different tools support architectural design and test case generation. Users of these tools want them to work together to fully support the user's design and development process. Thus tool integration is intended to produce complete environments that support the CMMI model based software development life cycle of maturity levels. There are five types of tool integration issues that must be addressed. These can be termed platform integration, presentation integration, data integration, control integration and process integration [14].

Service-Oriented Computing (SOC) arises as a new paradigm for distributing applications which envisions services as fundamental elements for developing applications [9].

So X-as-a-Service kind of pattern is what SOA is suggesting in terms of cloud computing terminology by adopting the basic principle SOA's loose coupling technique where the above five types of tool integration issues can be represented as SOA based services. Where X-as-a-Service can be replaced with Platform-as-a-Service (PlaaS), Presentation-as-a-Service (PraaS1), Data-as-a-Service (DaaaS), Control-as-a-Service (CoaaS) and Process-as-a-Service (Praas2) in cloud computing terminology that are available independently as loosely coupled services.

As design and development of SOA applications are inherently systematic [7], this is of particular importance for enterprises, when given their need to become more agile in order to react as quickly as possible to changing business environments and offer new services to customers, suppliers and partners that make a difference with respect to competition. SOAs can help to significantly reduce complexity at all levels. SOAs achieve their simplicity by following features: Decomposition-SOAs decompose large systems into application frontends and services; Appropriate granularity-The granularity of services is well suited to gain a high-level understanding of the entire system; Loose coupling by SOA architectural patterns can be Decoupled using technology-SOAs can be well understood without in-depth knowledge of technology; Reuse-SOAs result in the high level reuse of existing components; Documentation – SOA based services are well documented due to service contract to provide comprehensive understanding [6].

2. Platform Independent Integration Architecture

In order to get a web portal integration architecture that can be used in different domains, it is important first to design a platform-independent architecture [1]. The key to successful integration and interoperability lies in the intelligent use and management of metadata across all applications, tools and databases. Metadata management and integration can be accomplished through the use of the OMG's core MDA standards such as CWM, UML, MOF and XMI [10], where:

CWM stands for *Common Warehouse Meta-model* – It is a meta-model of the data model representing both the business and technical metadata that is most often found in the data warehousing and business analysis domains. CWMs are intended to be highly generic, external representations of shared metadata [10]. It requires the use of generalization and abstraction technique to translate the product-specifics to generics through standard extension mechanism making it compatible to CWM format [10].

UML stands for *Unified Modelling Language* – It is used for expressing in the Unified Modelling Language, which is an OMG standard language for modelling discrete systems by Rumbaugh. UML is the notational basis for the definition of CWM. Visual UML models are capable of automatic translation to other visual or non-visual formal languages to facilitate the support for interchange of CWM models in various platform and tool independent formats (e.g., XML) as well as the construction of tool-specific metadata from CWM models (e.g. translation of a CWM relational model into SQLDDL statements that actually build the schema) [10].

MOF stands for *Meta Object Facility*- It is an OMG standard defining a common, abstract language for the specification of meta-models. MOF is an example of meta-meta-model or model of the meta-model (also called as ontology). The MOF's support for the model life cycle semantics means that MOF implementation provides an effective metadata authoring and publishing tool, when combined with support for visual modelling. For example a fully MOF-compliant repository must provide a significant number of metadata services that as: persistence, versioning and directory services [10].

XMI stands for *XML Metadata Interchange (XMI)* **-** It is an OMG standard that maps the MOF to the W3C's eXtensible Markup Language (XML). XMI defines how XML tags are used to represent serialized MOF-compliant models in XML. MOF based meta-models are translated to XML Document Type Definitions (DTDs) and models are translated into XML documents that are consistent with their DTDs. XMI based interchange is so important in distributed and heterogeneous environments as the communication of content is both self-described and inherently asynchronous [10].

Object Management Group's Model-Drive Architecture (MDA) is an approach to system-specific and interoperability based on the use of formal models. In MDA, platform-independent models are initially expressed in a platform-independent modelling language such as UML. The platform independent model is subsequently translated into a platform specific model by mapping platform-independent models into some implementation language (e.g. Java) or platform using formal rules. The core standards of MDA such as CWM, UML, MOF and XMI form the basis for building coherent schemes for authoring, publishing and managing models within a model-driven architecture [10].

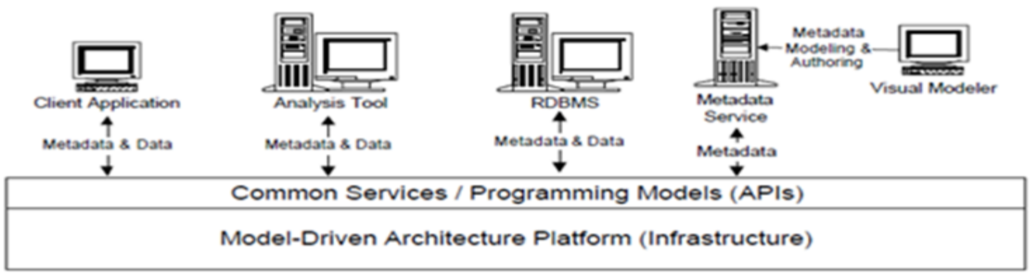

Client Application
Metadata & Data
Analysis Tool
Metadata & Data
RDBMS
Metadata & Data
Metadata Service
Metadata
Metadata Modeling & Authoring
Visual Modeler
Common Services / Programming Models (APIs)
Model-Driven Architecture Platform (Infrastructure)

Fig. 2.1 Example of a Realization of Model-Driven Architecture [10]

Metadata is critical to all aspects of interoperability within heterogeneous environment. In fact Interoperability is achieved by means of metadata [10], which is being used to provide system semantic definitions and capabilities facilitated in the form of standard APIs. Any MDA based system must have the ability to store, manage and publish both application and system-level metadata including descriptions of the environment itself.

The platform-independent architecture [1] (See Figure 2.2) aims to offer service-based platform independent architecture for web portal integration.

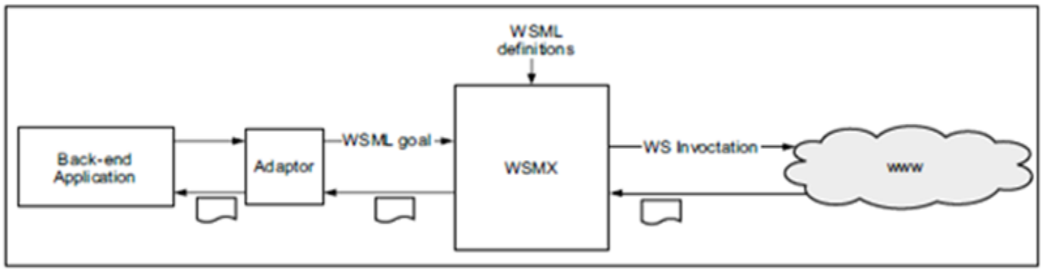


Fig. 2.2 External view of WSMX Architecture [1]

Web Service Execution Environment (WSMX) is a reference implementation for Web Service Modelling Ontology (WSMO) [1]. Its goal is to provide both a test bed for SMO and to demonstrate the viability of using WSMO as a means to achieve dynamic inter-operation of Semantic Web Services (SWS). The first version of WSMX provides the architecture needed for a middleware-based platform for integration, and as such it is concerned with dynamic discovery, mediation, selection and invocation and an implementation of these components.

2.1. Description of the Functional Usage of WSMX

Web Service Markup Language (WSML) descriptions of Web Services, ontologies, mediators and goals are sent to Web Service Execution Environment (WSMX) through Web Service Modelling Ontology (WSMO) editor for compilation. A back-end application creates a service requirement in a known source format and sends this to the WSMX adapter. The adapter takes the service requirement and translates it into a WSML message consisting of a goal that describes what a WSMX should execute. The goal is then sent to WSMX for execution. Before WSMX can execute the goal, WSML descriptions of the WS offering, the capability of matching the service requirement of the ontologies these WS use, and the source format ontology must have been created using User Interface (WSMO editor) are compiled to WSMX. When WSMX receives the WSML message with specific goal, it discovers the WS that best matches that goal, mediates the service requirement data following mapping rules between the source format ontology and the ontology of the discovered WS and finally invokes it, providing the data to it in the concepts and formats it expects [1].

Whereas metadata acts as control abstraction layer [5], the remaining g four generic and important layers from service based system integration perspective are Interaction, Process, Function and Data as against inclusion of Platform-as-a-Service for platform specific integration architecture (See Figure 2.1.1.).

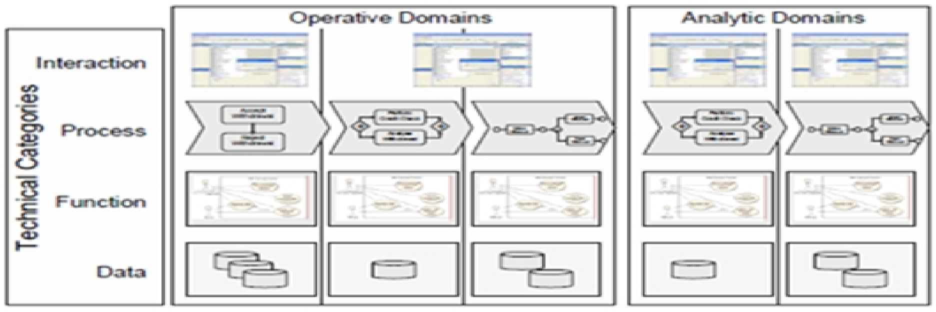


Fig. 2.1.1 Remaining Four Layers of Platform Independent Service Classification and Reference Schema [8]

3. Integrating Multi-Agent Systems (MAS) and Semantic Web Services (SWS)

The general idea of the approach to integrate MAS and SWS is given in Figures 3.1 & 3.2. For both Multi-Agent Systems (MAS) as well as Semantic Web Services (SWS), model transformations to the platform specific levels were provided by applying principles of model-driven development [4].

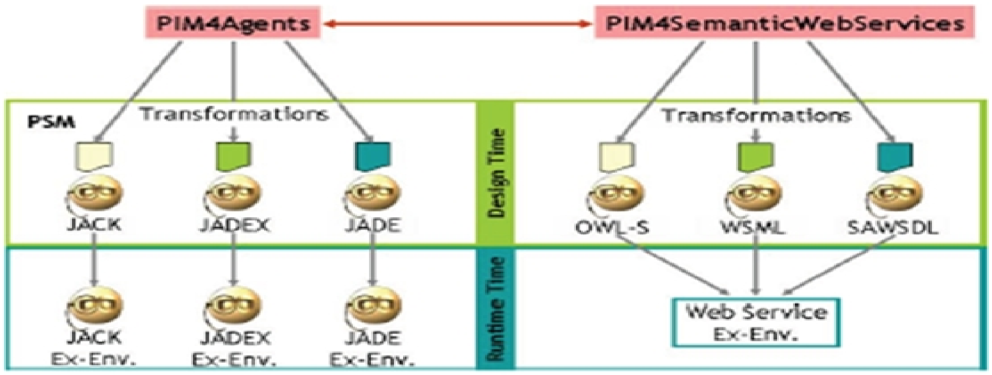


Fig. 3.1 Approach to Integration [4]

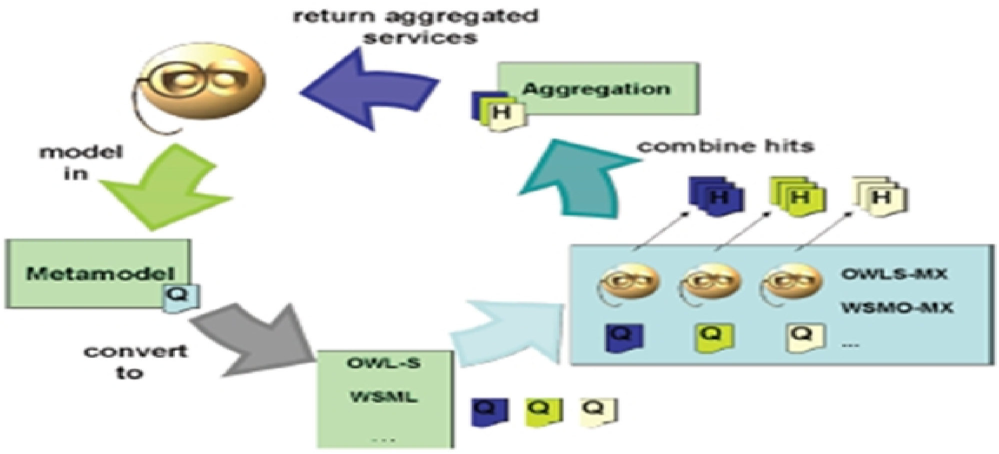


Fig. 3.2 Model-Driven Semantic Web Service Match Making [4]

This integration is based on a platform independent meta-model for agents and a platform independent meta-model for semantic web services.

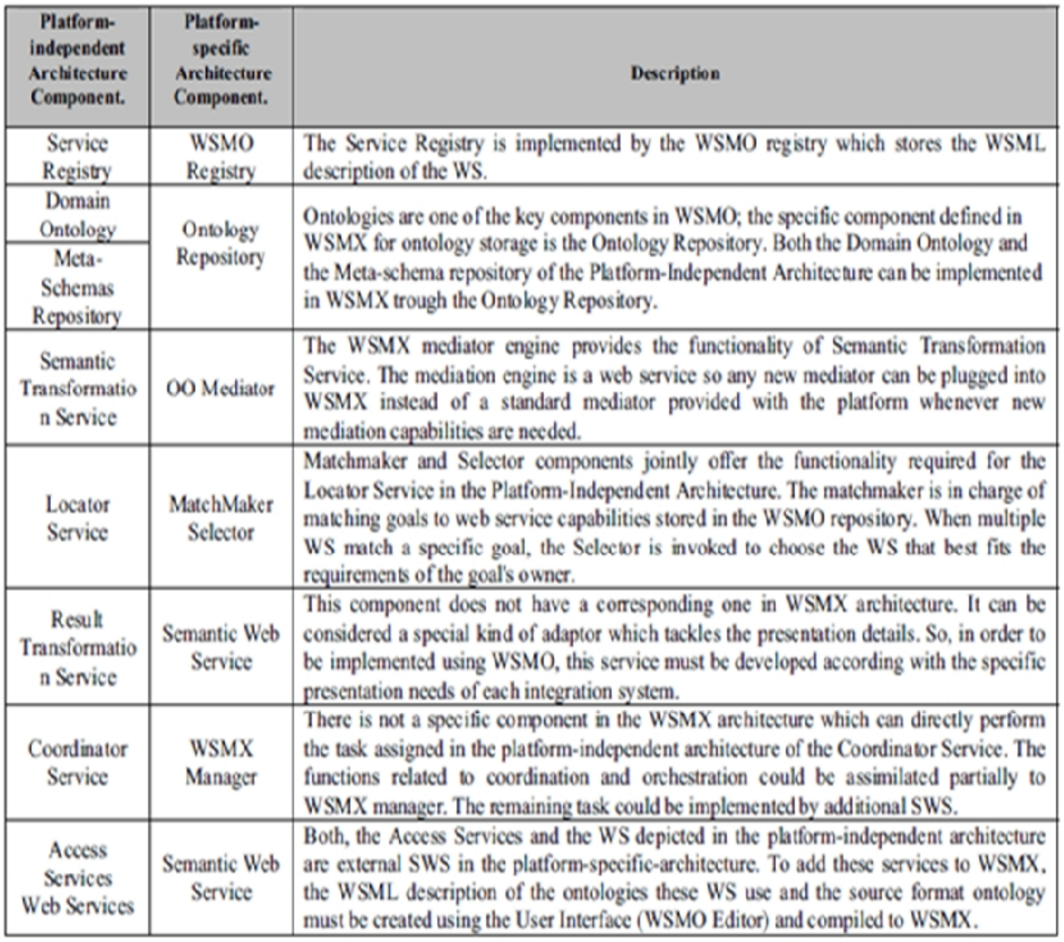
For Semantic Web Services, a model-driven semantic web services matchmaker agent is designed (See Figure 3.2) that discovers semantic services independent of selected description formats (OWL-S), (WSML) and (SAWSDL). The model-driven semantic service matchmaker (MDSM) performs an automatic service retrieval applying existing matchmakers (i.e. OWL-MX and WSMO-MX) for different formats and returns the most similar matches to the user [4].

4. PLATFORM-SPECIFIC INTEGRATION ARCHITECTURE

WSMO and OWL-S can be used as the specific platform to deploy the platform-independent service oriented architecture.

This section shows the specific deployment of the platform-independent architecture to a platform specific one using WSMO in the form of Table 4.1 that presents the transformation needed to deploy the platform independent architecture in the selected platform specific technology.

Table 4.1 Correspondence between platform-independent and platform specific architectures [1]



Platform-independent Architecture Component.	Platform-specific Architecture Component.	Description
Service Registry	WSMO Registry	The Service Registry is implemented by the WSMO registry which stores the WSML description of the WS.
Domain Ontology	Ontology Repository	Ontologies are one of the key components in WSMO; the specific component defined in WSMX for ontology storage is the Ontology Repository. Both the Domain Ontology and the Meta-schema repository of the Platform-Independent Architecture can be implemented in WSMX trough the Ontology Repository.
Meta-Schemas Repository		
Semantic Transformation Service	OO Mediator	The WSMX mediator engine provides the functionality of Semantic Transformation Service. The mediation engine is a web service so any new mediator can be plugged into WSMX instead of a standard mediator provided with the platform whenever new mediation capabilities are needed.
Locator Service	MatchMaker Selector	Matchmaker and Selector components jointly offer the functionality required for the Locator Service in the Platform-Independent Architecture. The matchmaker is in charge of matching goals to web service capabilities stored in the WSMO repository. When multiple WS match a specific goal, the Selector is invoked to choose the WS that best fits the requirements of the goal's owner.
Result Transformation Service	Semantic Web Service	This component does not have a corresponding one in WSMX architecture. It can be considered a special kind of adaptor which tackles the presentation details. So, in order to be implemented using WSMO, this service must be developed according with the specific presentation needs of each integration system.
Coordinator Service	WSMX Manager	There is not a specific component in the WSMX architecture which can directly perform the task assigned in the platform-independent architecture of the Coordinator Service. The functions related to coordination and orchestration could be assimilated partially to WSMX manager. The remaining task could be implemented by additional SWS.
Access Services Web Services	Semantic Web Service	Both, the Access Services and the WS depicted in the platform-independent architecture are external SWS in the platform-specific-architecture. To add these services to WSMX, the WSML description of the ontologies these WS use and the source format ontology must be created using the User Interface (WSMO Editor) and compiled to WSMX.

For Multi-Agent Systems, model transformations from the platform independent to the platform specific meta-models for JACK and JADE are defined as shown in the figure 3.1 [4].

It is interesting to note that there is another potential layer of service abstraction called Structure-as-a-Service (StaaS) that could be included by providing a subclass hierarchy of nested service categories, which serve as component types that can be applied to services. This structural perspective is complemented by the service connectivity. This structural perspective acts as reference architecture constraints from organisational and connectivity perspective on platform oriented runtime services (for e. g., See Figure 4.1).

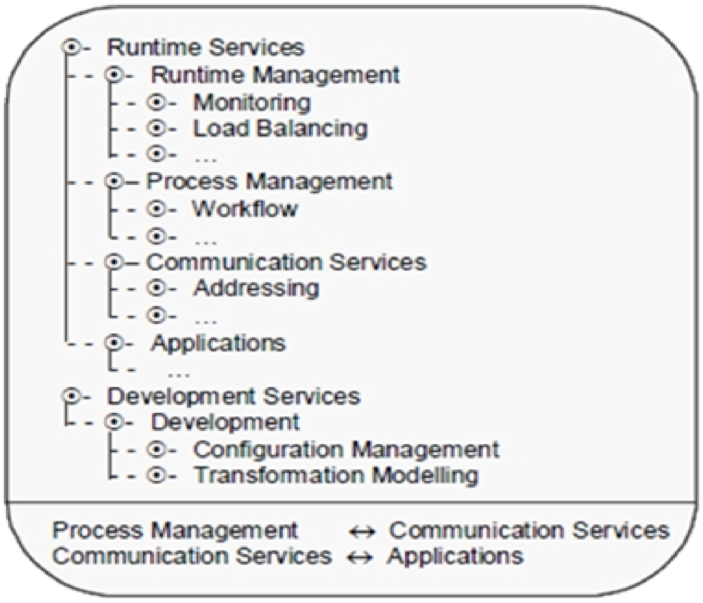


Fig 4.1 Structural Perspective as a reference architecture constraint–organisational and connectivity perspective on platform oriented runtime services [8]

5. CONCLUSION

Since the advent of the Internet, several works have gone a long way towards resolving the web integration problem. Our effort in this regard is to propose two kinds of architecture based solutions: (1) Platform Independent architecture based on service oriented paradigm for web portal integration and (2) Platform specific architecture. It is interesting to note that the following abstraction levels-as-a-service kind of pattern has been observed to be part of the cloud computing paradigm. They are: Structure-as-a-Service, Process-as-a-Service, Control-as-a-Service, Presentation-as-a-Service, Data-as-a-Service for Platform Independent Architecture Solutions and Platform-as-a-Service get added or included for platform-specific architecture.

REFERENCES

[1] Acuna, C. J., Marcos, E., Gomez, J.M. and Bussler, C. (2005). Toward Web Portals Integration through Semantic Web Services. Proceedings of the International Conference on Next Generation Web Services Practices (NWeSP'05), IEEE Computer Society.
[2] Erl, T. (2004). Service-Oriented Architecture: A Field Guide to Integrated XML and Web Services. Prentice Hall, NJ, USA, ISBN: 0131428985, Digital ACM Citation No. =983556.
[3] Erl, T. (2005). Service-Oriented Architecture: concepts, technology and design, Prentice. Hall.
[4] Hahn et al. (2008). Integration of Multi-agent Systems and Semantic Web Services on a Platform Independent Level. IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, IEEE Computer Society.
[5] Hedden, H. (2010). Taxonomies and Controlled vocabularies best practices for metadata. Jour-nal of Digital Asset Management, 6, pp.279-284.
[6] Krafzig, Banke and Slama (2005). Enterprise SOA- Service Oriented Architecture: Best Practices. Pearson Education Inc. Publication, USA.
[7] Natis, Y.V. (2003). Service Oriented Architecture Scenario. Gartner Research. ID No.: AV-19-6751, Gartner Inc, USA.
[8] Pahl, C., Hasselbring, W. and Voss, M. (2009). Service-Centric Integration Architecture for Enterprise Software Systems. Journal of Information Science and Engineering, 25, pp. 1321-1336.

[9] Papazoglou, M. P. and Georgakopoulos, D. (2003 October). Service-Oriented Computing. Communications of the ACM, Vol 46, No. 10, pp. 25-28.
[10] Poole, J. D. (2001). Model-Driven Architecture: Vision, Standards and Emerging Tech-nologies. In Workshop on Metamodeling and Adaptive Object Models, ECOOP 2001.
[11] Shiva Shankar, M. (2010). The importance of Enterprise Software Architectures. http://www.c-sharpcorner.com/uploadfile/shivamadishet/the-importance-of-enterprise-software-architectures/ Cited on 20-04-2012
[12] Vasista, T. G. K. (2012). Adaptive Enterprise: Design and Implementation Approaches for E-Government Integration, International Journal of Conceptions on Computing and In-formation Technology (IJCCIT), WAIRCO Publication, India (Accepted for Publication).
[13] Varma, V. A., Reklaitis GV, Blau, G. E., Pekny, J. F., (2007). Enterprise-wide modeling & optimization—An overview of emerging research challenges and opportunities. Computers and Chemical Engineering, 31, 692–711.
[14] Wasserman, A. I. (1990). Tool Integration in Software Engineering Environments. LNCS, Vol. 467, pp. 137-149, Springer Link Publications.
[15] Wiki (2012 January). Capability Maturity Model Integration. http://en.wikipedia.org/wiki/File:Characteristics_of_Capability_Maturity_Model.svg. Cited Jan 20, 2012.
[16] Choudhury, I., de Ceasare, S. And Florido, E. (2008) “An Object-Oriented Abstraction for Generic Enterprise Modeling”, International Journal of Enterprise Information Systems, Vol. 4 Iss. 1, IRMA International Publishing.
[17] Reinhartz-Berger, Iris and Dov Dori (2005), “A Reflective Metamodel of Object-Process Methodology: The System Modeling Building Blocks,” Business Systems Analysis with Ontologies, P. Green and M. Rosemann (Eds.), Idea Group: Hershey, PA, USA, pp. 130-173.

Authors

Dr. Al-Sudairi is currently working as an Associate Professor in College of Business Administration for MIS Department at King Saud University at Riyadh, KSA. He obtained Two Masters one in Economic and another in Business MIS from USA and a Doctorate on EDI and E-Business from University of Leicester, UK. His interested areas include IT, E-Business, E-Commerce, Information Systems Strategy and ERP. He has publications in reputed journals and conferences

Dr. TGK Vasista is currently working as Researcher at King Saud University, Riyadh, KSA. He obtained a Doctorate in Information Technology from St. Linus University, West Indies, a Master’s from University of Roorkee (Now called IIT-Roorkee), India And a Post-Graduate Diploma in E-Governance from University of Mysore, India. He Has former experiences in the field of IT as a programmer analyst in USA and as a senior Lecturer/Asst. Professor in academ ic field in the area of Systems and IT, E-Business and E-Governance.