ISSN: 2250–3676

[IJESAT] INTERNATIONAL JOURNAL OF ENGINEERING SCIENCE & ADVANCED TECHNOLOGY Volume-2, Issue-4, 946 – 950

SERVICE-ORIENTED MODELING – AN ANALYTICAL STUDY

Dr. Vinay Goyal[1], Amit Jain[2]

[1]*Director, Haryana Institute of Technology, Bahadurgarh (Haryana) INDIA,* ***vinayeq@yahoo.com***
[2]*Research Scholar, Teerthanker Mahaveer University, Moradabad (UP) INDIA,* ***amit_jainci@yahoo.com***

ABSTRACT

This research paper presents a deep insight into the analytical study of service-oriented modeling and its framework. Service-oriented modeling is a mechanism that will enable us to conceive software products that we have been developing, acquiring, and integrating during the past few decades as service-oriented constituents. Service-oriented modeling disciplines enable us to focus on modeling strategies rather than being concerned with source code and detailed programming algorithms. By employing this modeling paradigm, we raise the level from the grainy constructs of our applications, yet we must accommodate the language requirements of the reinforced platforms. Service-oriented modeling framework identifies two major modeling practices that drive and influence all modeling activities during a project or any business initiatives: abstraction and realization. The activities of SOM can always be repeated to perfect the resulting artifacts that the modeling disciplines require for delivery. To ensure the success of the service-oriented modeling process, modeling practices must be guided and enforced by modeling disciplines. The paradigm related to service-oriented modeling has been studied by keeping an eye on its role in the field of service-oriented architecture.

INDEX TERMS: SOM, Virtualization, SCMS, SOSA.

----------***----------

1. INTRODUCTION

As human beings, we are ardent to innovative imaginings that undertake to alter our lives and create new opportunities. We also tend to rapidly replace old technologies with new ones. Ours is a versatile society that runs on tomorrow's software piled on top of the technology layers of yesterday and today.
These futuristic software concepts would probably contribute yet another layer to our already complex computing environments, one requiring resources to maintain and budgets to support. This layer would sit on top of technological artifacts accumulated over the past few decades that are already difficult to manage.

Not long after the new millennium, discontent over interoperability, reusability, and other issues drove the software community to come up with the service-oriented architecture (SOA) paradigm. Even readers who are not familiar with SOA will probably agree that it is rooted in traditional software development best practices and standards. To fulfill the promise of SOA, superb governance mechanisms are necessary to break up organizational silos and maximize software asset reusability. The SOA vision also addresses the challenges of tightly coupled software and advocates an architecture that relies on the loose coupling of assets. On the financial front, it tackles budgeting and return-on-investment issues. Another feature that benefits both the technological and business communities is a reduction of time to market and business agility. Indeed, the list of advantages continues to grow.

It is possible that SOA may fail to deliver on its assurance, but if it does, we, business and IT personnel, must shoulder some of the blame. SOA may turn out to be little more than a technological fine drill, if we are ambivalent about the roles and responsibilities of legacy software in our existing and future organizational strategies; if we fail to tie together past, present, and future software development initiatives; and if we disregard the contributions of previous generations of architectures to today's business operations, then the time is not so far when there will be huge losses faced by IT industry in terms of production and revenue generation. Indeed, the idea of properly bridging new and old software technologies is a novel one.

Service-oriented modeling is a mechanism that will enable us to conceive software products that we have been developing, acquiring, and integrating during the past few decades as service-oriented constituents. These entities - either legacy applications written in languages such as COBOL, PL1, Visual Basic, Java, C++, C#, or diverse empowering platforms and middleware —should all take part in an SOA modeling framework. Most important, they should be treated equally in the face of analysis, design, and architectural initiatives, and should simply be recognized as services. This SOA modeling approach is well suited to provide tactical, short-term solutions to enterprise concerns.

2. CONCEPT & DESCRIPTION

Service-oriented modeling is a software development practice

that employs modeling disciplines and language to provide strategic and tactical solutions to enterprise problems. This anthropomorphic modeling model advocates a holistic view of the analysis, design, and architecture of all organizational software entities, conceiving them as service-oriented assets, namely Services. Modeling activities are typically embedded in the planning phase of almost any project or software development initiative that an organization conducts.

The modeling paradigm embodies the analysis, design, and architectural disciplines that are being followed during a slotted software development activity. These major modeling efforts should not center only on design and architectural objects such as diagrams, charts, or blueprints. Indeed, modeling deliverables is a big part of a modeling process. But the service-oriented modeling venture is considerably about imitating the real world. It is also about visualizing the final software product and envisioning the coexistence of services in an interoperable computing environment. Therefore, the service-oriented modeling paradigm advocates first creating a small facsimile of the "big thing" to represent its key uniqueness and activities — in other words, Design undersized, Obtain vast Analyze little, Accomplish gigantic!

The service-oriented modeling process begins with the development of a minuscule model on a document. This may involve modeling teams in whiteboard analysis, design, and architecture sessions, or even the employment of software modeling tools that can visually exemplify the solutions arrived at.

Thus, the recreation process entails the creation of a virtual world in which software constituents, interface & collaborate to provide a viable therapy to an organizational apprehension. The hopeful advances in service orientation encompass productive vision for the technologies that will influence the Internet over the subsequent years and the innovative functionalities they will involve. [4]

3. DISCIPLINES IN MODELING

Creating a diminutive copy of a final software product and its supporting environment can obviously reduce investment risk by ensuring the success of the imminent software development scheme. This can be achieved by employing analysis, design, and architectural disciplines that are driven by a modeling strategy that fosters asset reusability, a high return on investment, and a persuasive value proposition for the organization.

Service-oriented modeling disciplines enable us to focus on modeling strategies rather than being concerned with source code and detailed programming algorithms. By employing this modeling paradigm, we raise the level from the grainy constructs of our applications, yet we must accommodate the language requirements of the reinforced platforms. We focus on identifying high-level business and technological asset reusability and consolidation opportunities, but we must also foster the reuse of software building blocks such as components and libraries. We austerely look for for interoperability solutions that can conduit heterogeneous technological environments, but we also concentrate on assimilation and message exchange accomplishment aspect.

By producing a small version of the final relic, we are also engaging in a learning and verification process. This activity characteristically would enable us to validate the hypothesis that we have made about a software product's capability and its ability to operate flawlessly later on in a production environment.

We were also being given the opportunity to inspect key aspects of software behavior, examine the relationships between software components, and even understand their internal and external structures. We are involved in a software assessment process that validates the business and technological motivation behind the construction of our tangible services.

To better understand, the key characteristics of a future software product and its environment, the assessment effort typically leads to a proof-of-concept, a smaller construction project that concludes the service-oriented modeling initiative.

4. SOM STANDARDS

Service-oriented modeling principles capitalize on devised SOA standards already in use by organizations and professionals. These are best practices that are designed to foster strategic solutions to address enterprise concerns, and to overcome the short-sightedness that is frequently attributed to organizational tactical decisions.

We have arrived at the following modeling principles, which promote business agility, software asset reuse, loosely coupled service-oriented environments, and a universal modeling language that can address software interoperability challenges. These are virtualization, metamorphosis and literate modeling.

ISSN: 2250–3676

[IJESAT] INTERNATIONAL JOURNAL OF ENGINEERING SCIENCE & ADVANCED TECHNOLOGY Volume-2, Issue-4, 946 – 950

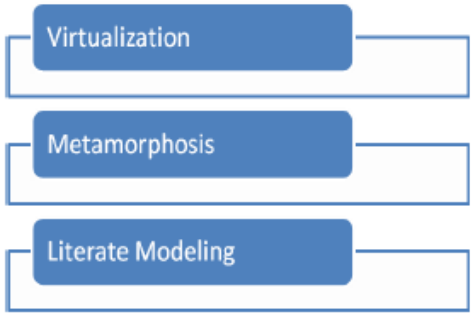

Fig-1: SOM Standards

5. VIRTUALIZATION

Modeling software is essentially a process of manipulating intangible entities. These are typically nonphysical assets that reside in peoples' minds or appear on paper. An effective modeling process should be as visual as possible, enabling business and technology personnel to view software elements as if they were concrete assets. [1]

The virtual and reality aspects of our surroundings have been debated by numerous philosophers going back to the eighteenth century. The traditional statement that everything has a reality and virtuality or everything other than, what is virtual is a reality are in agreement with sociologist, philosopher, and information technology community claims that virtuality is the focal point of software design.

This world as of now is consisting of two main elements: (a) the landscape that "fastens" all pieces together; meaning the environment that empowers and executes services and (b) the services that communicate, interact, and exchange information to provide business value. Moreover, a virtual world can effectively imitate a heterogeneous computing landscape by treating software assets as equal partners in a modeling endeavor.

The visualization process that we pursue enables us to model relationships, structures, and behaviors of services that would provide satisfying solutions to organizational problems. These goals can be achieved by fostering asset reusability, promoting a loosely coupled computing environment, and resolving interoperability challenges across organizations.

6. METAMORPHOSIS

The business environment that we all share is a self-motivated market that keeps sprouting and altering direction and also influences technological trends and application development. This vibrant business landscape often dictates alterations to a service's behavior, structure, and relationship to its environment during its life span. These modifications typically start at a service's inception, when it manifests as an indefinable entity — an idea — and then continue as the service evolves into a physical software asset that executes business functionality in production. This transformation process is the essence of the metamorphosis paradigm driven by service-oriented modeling disciplines that ensure software elasticity, and ultimately, business agility. But the transformation process does not stop with the deployment of services to production.

Imagine how frequently a valuable application is involved in multiple project iterations that lead to software upgrades. Think about the numerous instances of a service finding its way back to the drawing board to be redesigned and then transmitted back to the production environment again. This is typical of the software development life cycle, during which software products are upgraded, enhanced, and redistributed.

7. LITERATE MODELING

Should a programming language offer a simple syntax that is easy to understand, and is instinctive and decipherable? Or should it offer formal grammar rules, and symbols that only developers can handle? Should a programming language be based on machine-readable source code that is easy to debug and optimize? Or should it be considered a scientific artifact, a mathematical formula that only experts on the subject can deliver?

This long-running discussion is believed to have begun in the early 1980s and encompasses two major approaches to software development that have major ramifications for the service-oriented modeling paradigm. The first relates to the Knuth's theory of "literate programming" in which he argues: "I believe that the time is ripe for significantly better documentation of programs, and that we can best achieve this by considering programs to be works of literature. Hence, my title: 'Literate Programming'."

The second approach was introduced by Dijkstra in his 1988 article "On the Cruelty of Really Teaching Computer Science," in which he claims that programming is merely a branch of mathematics. He writes: "Hence, computing science is — and will always be — concerned with interplay between mechanized and human symbol manipulation, usually referred to as 'computing' and 'programming' respectively." [2]

This discussion further relates to modeling paradigm. Should service-oriented modeling be founded on a specific programming platform structure that only developers and modelers can utilize? Or should modeling disciplines offer universal and easy to understand notations that are independent of language? Should a service-oriented modeling approach be tied to fashionable technologies? Or should a modeling language offer tools to design and architect multiple generations of legacy

ISSN: 2250–3676

[IJESAT] INTERNATIONAL JOURNAL OF ENGINEERING SCIENCE & ADVANCED TECHNOLOGY Volume-2, Issue-4, 946 – 950

platforms, applications, and middleware?

The service-oriented anthropomorphic modeling approach provides easy mechanisms to address analysis, design, and architectural challenges and perceives software assets as having human characteristics. In the virtual world that we are commissioned to create, services "interact," "behave," "exchange information," and "collaborate;" they are "retired," "promoted," "demoted," and "orchestrated." Inert software entities are often treated as though they had human qualities. This "literate modeling" approach obviously enhances the strategies that are pursued during business initiatives and projects.

8. ORGANIZATIONAL SOSA

The service-oriented modeling paradigm regards all organizational software assets as candidates for modeling activities. We not only conceive them as our service-oriented modeling elements, meaning services , but we also evaluate them based on their contribution to a service-oriented environment, in terms of integration, collaboration, reusability, and consumption capabilities.

These assets are also subjected to the modeling discipline activities. They are the enduring artifacts of the service-oriented modeling process and are regarded as units of concern, discovery, analysis, design, and architecture in a business initiative or a service-oriented project. The illustration that the various service-oriented software assets can be involved in providing solutions to organizational concerns: concepts, foundation software, legacy software, repositories, and utility software. [3]

9.PRACTICES OF SOM

Organizational empowering middleware and platform products are the basic software ingredients of the service-oriented modeling practice. Middleware products offer integration, hosting, and network environment support, including message orchestration and routing, data transformation, protocol conversion, and searching and binding capabilities.

This software asset category may include application servers, portal products, software proxies, SOA intermediaries, gateways, universal description, discovery and integration (UDDI) registries, and even content management systems. Message-oriented middleware (MOM) technologies are also regarded as middleware, which may include traditional message busses or enterprise service busses (ESBs). Conversely, software platforms are akin to frameworks that enable languages to run. This category may also include operating systems, runtime libraries, and virtual machines.

CONCLUSION

As SOA modeling is important to the service-oriented system development, this paper highlights the describing problem of SOA within the context of the SCMS and proposes a modeling framework as solution.

However, there are some possible extensions could be made on the method defined here in future. Firstly, the behavior of service component will be taken into account and appropriate specification will be defined for them. Secondly, the meta-model definition will be extended so that it enables proving properties and reasoning about specifications. Thirdly, some modification will be made to specify dynamic configuration and dynamic reconfiguration as well as service components composing and recomposing in a service-oriented system.

REFERENCES

[1] www.en.wikipedia.org/wiki/Virtuality

[2] Edsger W. Dijkstra, ― On the Cruelty of Really Teaching Computer Science, December 1988, p. 16.

[3] Journal of Digital Information, Vol. 5, Issue 1, Article No. 298, July 16, 2004.

[4] Dr. Vinay Goyal, Amit Jain – Service-Oriented Computing & its framework, published in International Conference on Resurging India- Myths and realities, TMU, Moradabad, and March 2012.

ABOUT THE AUTHORS:

DR. VINAY GOYAL

Internationally recognized expert in Software Engineering and Academician and has more than 15 years experience, published numerous papers in this field. His current research interests include Soft computing Techniques, Data Compression, Software Re-Engineering and Mathematical Modeling. He is presently, designated as Director, Haryana Institute of Technology, Bahadurgarh (Haryana) India. He has two books in his credit.

AMIT JAIN

Is a Doctoral Researcher at Department of Computer Application at Teerthanker Mahaveer University, Moradabad (Uttar Pradesh) India. He has around 10 years of teaching experience. He is currently working on Legacy Systems, Re-Engineering, Software Evolution and Software Migration. His

research interest includes Service-oriented architecture, Service-oriented computing.