ENHANCEMENT OF Y-MODEL FOR SERVICE-ORIENTED MODEL

Varun Kumar Rajpoot*

Pragya Siddhi*

Dr. Anurag Singh Baghel**

Abstract: Service-Oriented Architecture (SOA) has received significant attention. In SOA, services are discovered and composed according to the goal specification. SOA can rightly be called an improvement over Component Based Software Engineering (CBSE), however it still uses CBS technologies for development of services. We aim to present a novel process for development, publishing and use of services. In addition, this paper presents a hypothetical process for composition of these services with more complex services, to cater to the end user.

Keywords: Component-Based Software Development (CBSD), Service-Based Development.

*M.Tech (Software Engineering), Gautam Buddha University, Greater Noida

**Phd, Assistant Professor, Gautam Buddha University, Greater Noida

1. INTRODUCTION

Component-Based Development (CBD) is older better-defined, and better –known in the software engineering environment. In the past a few years, rapid progress has been occurred in the field of Service-Oriented Development (SOD), which represents a paradigm shift from Component-Based Development (CBD) to the Service-Based Development. "Figure 1" shows Component-Based Development.

The Service-Oriented Computing (SOC) paradigm refers to the set of concepts, principles, and methods that represent computing in Service-Oriented Architecture (SOA) in which software applications are constructed based on independent component services with standard interfaces (*W.T. Tsai, et al. 2005*).

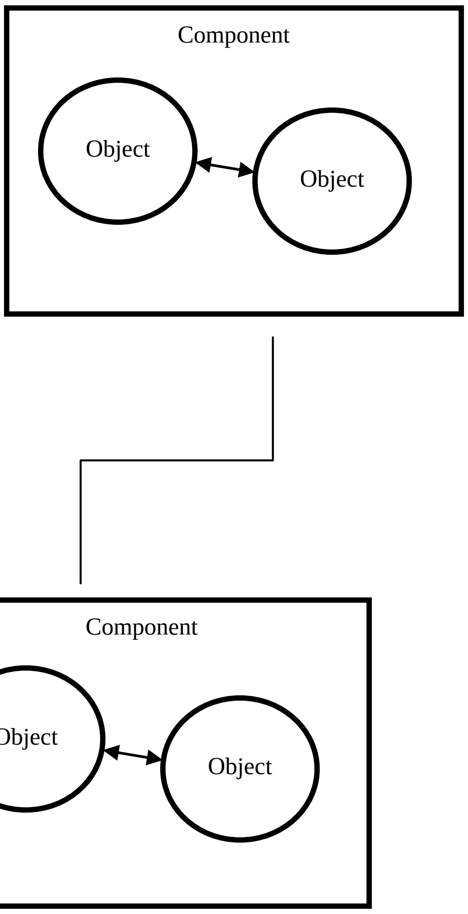

Fig 1: Component-Based Development

Service-Oriented Development is evolution of Component Based Development, Interface based Design (Object-Oriented) and Distributed Computing. "Figure 2" shows Service-Based Development.

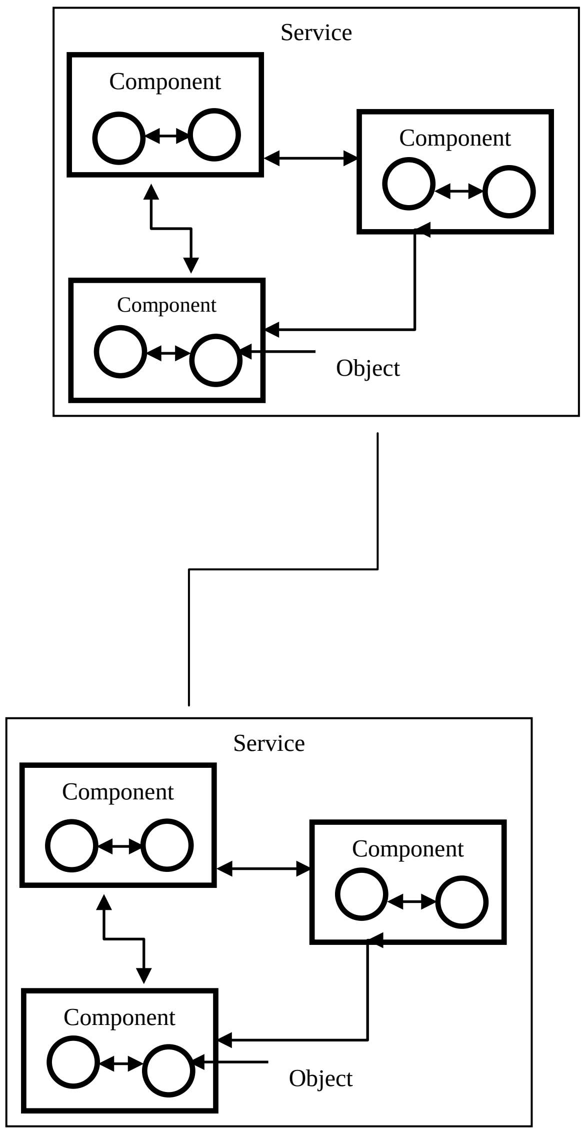

Fig 2: Service-Based Development

1.1 **SOA Entities:**Service-Oriented Architecture is an architectural style that defines an interaction model between three main functional units: the application builders (also called service requestors), the service brokers (or publishers), and the service developers (or providers) as shown in "Figure 3" in which the consumer of the services interacts with the service provider to find out a service that matches its requirement through searching registry (Y.H Wang, et al. 2011).

1.2 **Service Provider:**Service provider is an entity network addressable. It can accept and perform the request from service consumer, and service provider can make request by using mainframe, service or other software. It provides the definite service description and the implementation of the services. Theservice provider can be a component, or other type ofsoftware systemthat fulfils the service consumer's requirements.

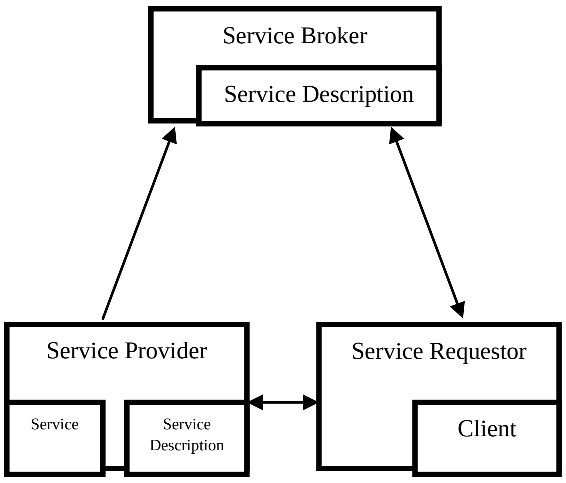

Fig 3: SOA Entities

1.3 **Service Broker:**Service broker is similar to service dictionary, which has Universal Description, Discovery and Integration (UDDI) function, responsible for receiving and storing the contract from service provider, then provides contract to service consumer which need it.

1.4 **Service Consumer/Application Builder:**Service Consumer is the entity in SOA that looks for a service to execute a required function. Service consumer can be end-user, application or some type of software module that needs the services. It will be looking for suitable service in service registry, then confirm the location of service, bind service, and perform the function of service. Service performs services on the basis of request and response on the basis of contract.

2. BACKGROUND

The development of service-oriented system faces many challenges, as stated by SOC research studies. Many question need to be answered, such as:

- Which activities should be considered for the development of the service-oriented system?
- How to make modern service-oriented system from legacy system?
- How to attain suitable services from business process?

In this context, various methodologies have been proposed to demonstrate the lifecycle of the service-oriented development. None of the existing service-oriented methodologies cover all issues of the service-oriented development; they are only relevant to specific aspects of the service-oriented development (M.F.Gholami, et al. 2010).

Model proposed by (Y.Kim and H.Yun2006) is based on identifying services from the legacy system. These models have lack of full coverage of the service-oriented development cycle, lack of supportive documents on their practical use.

(S. Moosavi et al. 2009) proposed a method for service-oriented design which is based on identifying different types of variations then establishing variability model for establishing adaptable services.

Luiz Fernando's Y model is most widely used for the development of component based software. Objective of this thesis is to adapt the existing Y-model to Service-Oriented Development to achieve the following:

- To cover most of development activities of service provider, service consumer and service broker.
- Legacy system adaption and reusability to build individual service and complete application.
- Service Analysis
- Variation Analysis in the service.
- SWOT analysis.
- To consider communication among components.
- Service-Oriented testing to test services.
- Quality assurance after integration.

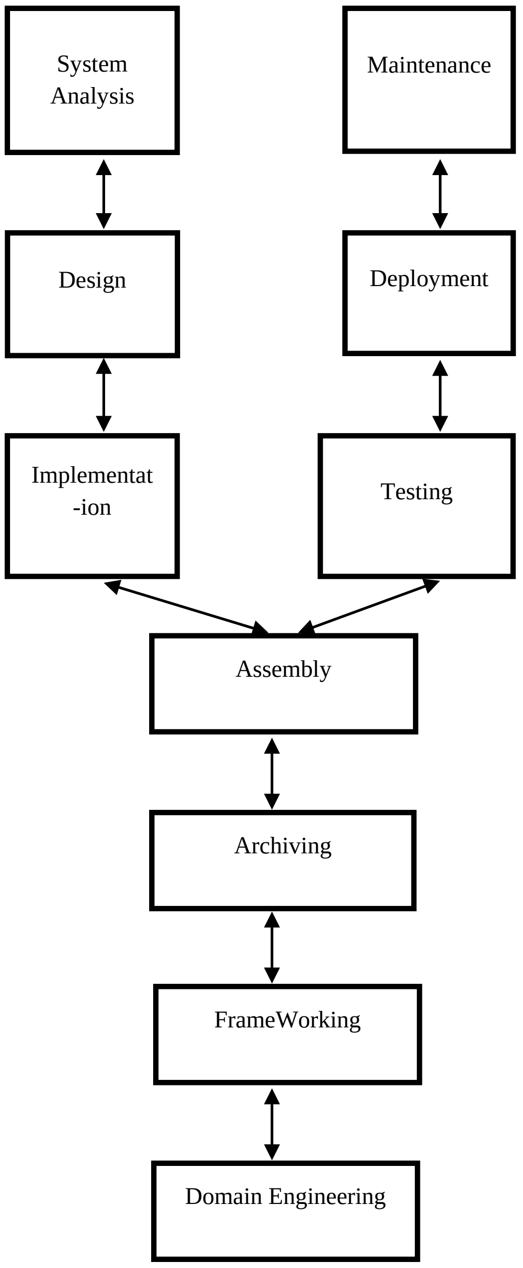

Fig 4: Y-Model for Component-Based Development

3. ENHANCED Y-MODEL

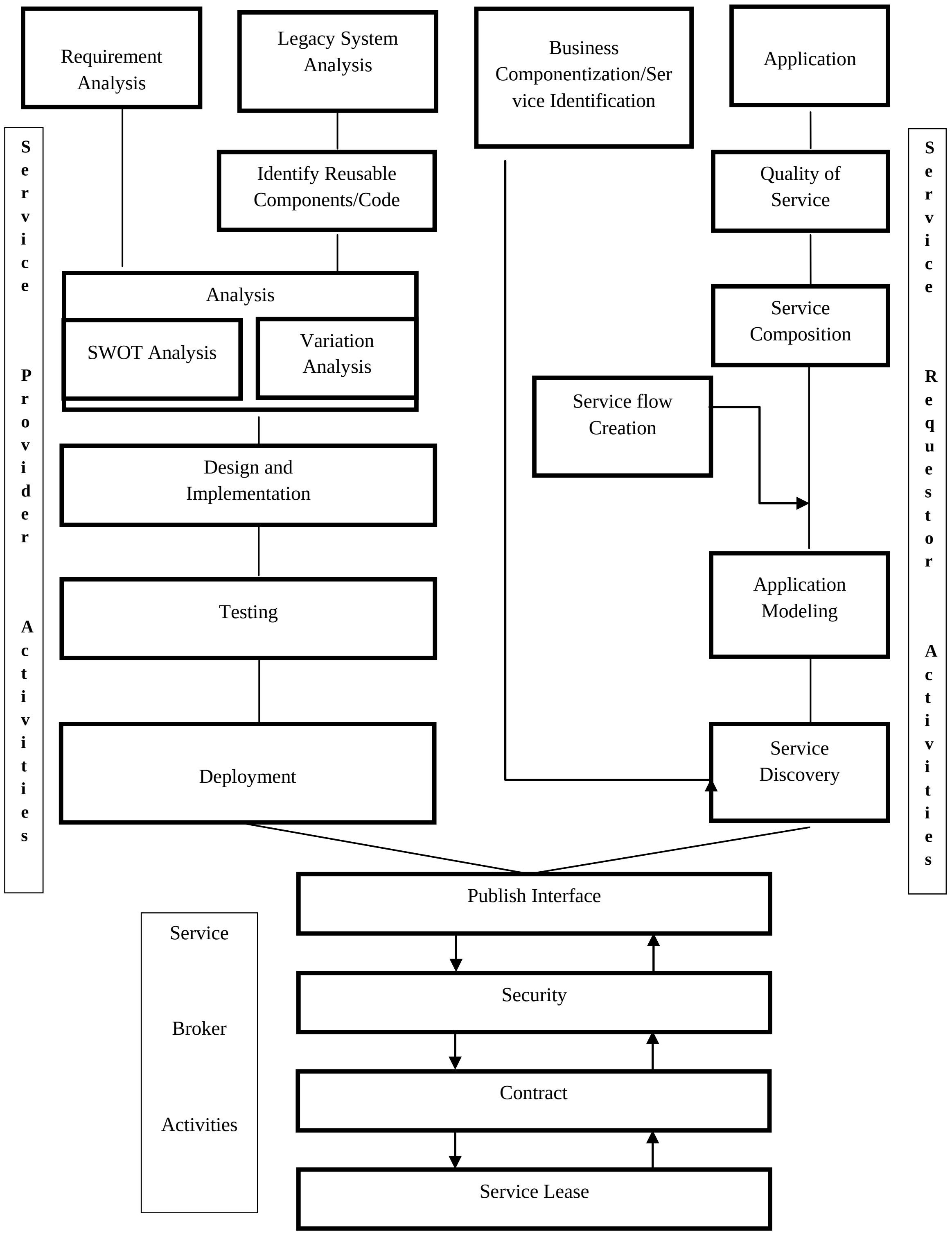

Fig 5: Enhanced Y-Model

3.1 **Service-Provider Activities**

Service Provider develops the loosely coupled services and these services are deployed on the provider sides.

*3.1.1**Requirement Analysis***: Service-Oriented Requirement Engineering (SORE) is different from traditional requirement. SORE shares some activities with the traditional requirement engineering but its focus is to identify reusable services. Before starting requirement analysis some design should be made. Some of the SORE features are:

3.1.1.1 Reusability-orientation: SOA emphasize on reusability. Once item is published, it can be reused by other.

3.1.1.2 Domain Specific: Most requirement process/techniques are often domain independent, as they are often application to a wide range of applications and domains.

3.1.1.3 Model-Driven Development: SORE assumes a model-driven approach. In which series are modeled using modeling language. UML, BPMN are used as modeling language.

In the beginning informal and unstructured requirements exists. Unstructured and informal requirements are translated into structured and formal model.

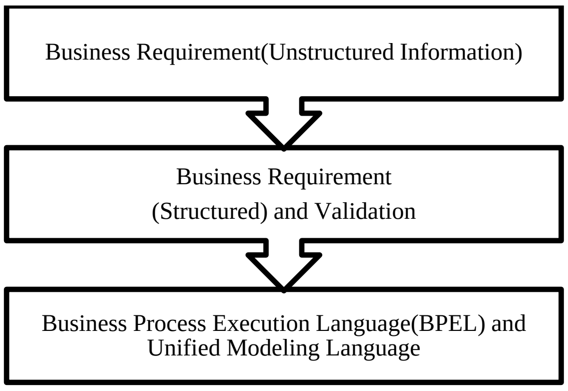
Business Requirement(Unstructured Information)
Business Requirement (Structured) and Validation
Business Process Execution Language(BPEL) and Unified Modeling Language

Fig6: Service Requirement Modeling

After requirement analysis model is transformed into Unified Modeling Language (UML) or Business Process Modeling Notation (BPMN) to start design and integration activities is shown "Figure 6".

In the detailed study of legacy system Use Case Model help to identify reusable services. From legacy system, reusable services or reusable module are identified.

3.1.2 **Legacy System Analysis:**A service is implemented by complex collaboration of many components or objects. Legacy systems are one of the most valuable software assets to reduce development time, cost, and effort and to increase reliability. In the legacy system analysis, Use Case Model helps to identify unit service or components for reuse and in analysing workflow of reusable units. In legacy system analysis, reusable components or code are identified.

3.1.2.1 Reusable Component Identification:After breaking up a service into components reusable components are found out and their dependency is analyzed.

3.1.2.2 Reusable Code Identification:Aspect oriented programming and program slicing helps to find out reusable code from a service or component. Identified code can be used in the implementation of a service.

3.1.3 **Analysis Activity:**In proposed model we introduced two type of analysis before starting design activity, which are as follows:

3.1.3.1 SWOT Analysis:Same service can be developed by different service providers. One can have higher probability to compete with other. Here 'S' stands for Strength, 'W' stands for Weakness, 'O' stands for Opportunity and 'T' stands for Threat. Opportunity is the area which attracts the enterprise, in the area, enterprise owns competition to use the services. Threat means the challenges formed by unfavorable development trend in environment. Every enterprise should periodically check its strength and weakness. Strength means one service is better with other service.

3.1.3.2 Variation Analysis:Services are not created for predefined users but are created for many unknown users and to adapt new concepts. Service variation can exist in business process or in unit service; variation can exist in workflow, composition, interface and logic (Chang H. S. et al., 2007). On the basis of variation analysis variation model is identified for establishing adaptable services. Some basic variations are:

3.1.3.2.1 Workflow Variability:A business process is carried out in a sequence of unit services, called workflow. In a workflow, some unit service may not be requested to a particular service user and some part of the sequencemay be differentlyperformed to other particular service user. Thus, a service (or a part of service) can perform in different ways for different users.

3.1.3.2.2 Composition Variability:A business process composes several unit services to fulfill the end users' requirements. For one unit service in the workflow, there may be more than one possible service interfaces which implement the service with different implementation logics or quality attributes. In this case, variation occurs on selecting the most appropriate service interface. That is, depending on user requirements, different service interfaces of the implementation can be composed and it is called Composition Variability.

3.1.3.2.3 *Interface Variability*:It occurs when the interfaces of unit services do not match to the interfaces of services published in UDDI service registry. A unit service is a logical unit of service that is composed into various business processes. The services registered in a service registry are published and available services, of which interfaces are typically derived from implemented service components and interface specifications of some standards.

3.1.3.2.4 *Logic Variability*: A service component includes operations for providing the functionality of unit services. The service component should provide different logics depending on particular requested services. It is called Logic Variability.

3.1.4 **Design and Implementation:**On the basis of requirement analysis, identified reusable services and variation model, Design activity is started. UML, BPMN, BPEL and Use Case Model helps in design activity. These services can be implemented in any language like C, C++, Java etc.

3.1.5 **Testing of the Service:**Traditionally in component-based software system three types of testing are performed: unit testing, integration and system testing. Service provider performs unit testing, service testing, integration testing and system testing for testing services. For service-oriented integration unit testing and group testing of interface is performed.

3.1.5.1 Unit Testing:Unit testing in SOA can be carried out just like in Component-Based Software Development. In unittesting individual unit of code is tested. Inunittesting focus is on functional correctness and hence correct implementation of algorithm. Unit testing can be performed as white-box and black-box, manual and automated, code-based and model-based.

3.1.5.2 Service Testing:Integration testing of SOA is different from traditional testing. In SOA new layer of testing service testing is performed. In service testing, there is less focus on correct implementation of algorithms but on the integration of the functional units inside the component (or service).

3.1.5.3 Integration Testing:Services are more loosely coupled than components. In contrast to the CBS approach, integration testing cannot rely on homogeneous components with tightly connected interface, therefore adaptability and distribution of SOA demands additional consideration for integration testing. Especially the effect of message racing and its implication have to be considered during system development and should be tested thoroughly. Message racing in this context refers to the situation where messages are not received in the same order as they were sent.

3.1.5.4 System Testing:Service testing is based on high-level usage scenario and business requirements that have been defined by business analyst or customers. Use interface testing is most appropriate to carry out system testing as the system will be carried as a whole. Generally system testing is performed after the deployment of the service on the server.

3.1.6 **Service Deployment:**The goal of this activity is to make the system available for use. Services are deployed in an operational environment on the service provider side. In this environment service become available to service consumer. Services deployment location is transparent. Service consumer does not have any need to know the location of the service. Universal Description, Discovery and Integration (UDDI) protocol is added with the deployed service to make it discoverable for service consumer. System testing is performed after deployment to check service is working correctly or not.

After the deployment of the service broker activities start to make the market for the services.

3.2 **Service Broker Activities:**

Service broker performs third party role in SOA. Broker is intermediary between service provider and consumer. Service Broker uses a repository (or registry) where serviceproviders publish interfaces of services. These interfaces are queried by consumer to meet there required. A service broker provides contract, security protocols and data format (incoming data and outgoing data).

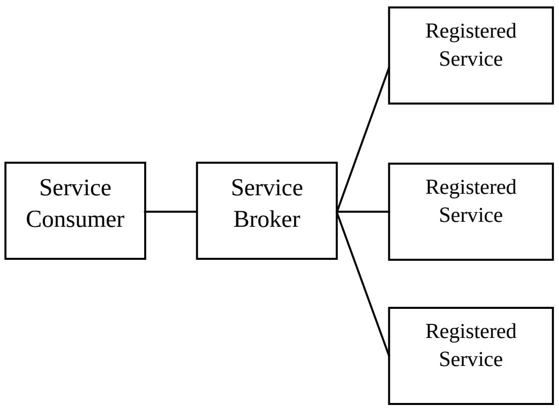

Fig 7: Service Broker Registered Services

A broker can publish services of different service providers and a consumer can invoke services of different brokers. The broker may invoke one or many services concurrently depending on how it is configured. If many services are invoked, it may wait for all to complete or just one to complete before notifying the client, if running synchronously.

3.2.1Security:Service broker provides security to service consumer. Security allows consumers to reach their application securely, without having complex back end configuration.

There is a lot of personal information and potentially secure data that people store on their computers, and this information is now being transferred to the service. This makes it critical to understand the security measures that their service provider has in place, and it is equally important to take personal precautions to secure their data.

The first thing consume must look into is the security measures that their service provider already has in place. These vary from provider to provider and among the various types of services.

- What encryption methods do the providers have in place?
- What methods of protection do they have in place for the actual hardware that consumer's data will be stored on?
- Will they have backups of my data?
- Do they have firewalls set up?

Many service providers have standard terms and conditions that may answer thesequestions. A smallbusiness user may have slightly more room to discuss the terms of their contract with the broker and will be able to ask these questions during that time.

There are many questions that consumer can ask, but it is important to choose a service provider that considers the security of service consumer's data as a major concern.

Since the processing takes place on the server and there is no hard drive, there is less chance of the malware invading the device. Also, since thin client do not work without a server, there's less chance of them being physically stolen.

3.2.2 Service Contract:A service contract is an interface that defines the message type used by service provider and service consumers to exchange message (Skonnard, 2005). Service contracts specify one or more operations that represent individual message exchange or a request/reply message exchange. Service broker provides contract to service consumer provided by service provider.

The service contract attribute defines the service interface and the operation contract attribute specifies methods of the services. A service contract is converted into Web Service Definition Language (WSDL) definition.

A web definition file has the following two parts:

1. A technology-independent abstract description that describes the service interface (port Type), its methods (operations) and their input-output parameters.
2. A concrete description for binding the communication protocol with the port (end point) of the service.

3.2.3 Service Lease:Service lease specifies the amount of the time that a service contract is valid. Service broker grants it to consumer, only in this duration a consumer can request for a specified service. When the lease runs out, the consumer must request a new lease from the registry. The lease is necessary for services that need to maintain state information about the binding between the consumer and provider. The lease defines time for which the state may be maintained. It also further reduces the coupling between the service consumer and the service provider, by limiting the amount of time consumers and providers may be bound.

After publishing of the services they are ready to be used by service consumer.

3.3 **Service Consumer Activities**

Service consumer invokes the published services on the network on the basis of contract and lease. First of all serviceconsumer identify the service to adaptinto theapplication. Activities performed by the service consumer are as follows.

3.4**Service Discovery:**After componentization service consumer discovers the service on the network after querying service repository with required characteristics. Discovery can be done in various domains. Service discovery and service composition is done by syntactical matching of Web Service Description Language (WSDL) or by semantic matching of Web Service Description Language (WSDL). Two basic domains are:

3.4.1 StructuredDiscovery Approach:Structure discovery is a syntactical approach to find out services which is based on interface description and the definition of the data (Input/Output) message exchanged between the communications partners. In structured discovery WSDL are matched to check that input/output message of service description are semantically linked via inheritance relationships or not.

3.4.2 Lexical Discovery Approach: Lexical discovery approach use natural language description. Web service operation names usually contain some terms describing their functionality. WSDL and other standards allow embedding natural language description. Lexical algorithms remove stop words from those descriptions; find synonyms using lexical databases like WordNet.

3.5 **Application Modeling:**After business componentization services are identified and after finding required services process is modelled. Business processes should be modelled and optimized in order to map appropriately to the services that satisfy business goal. During the modeling phase, service dependencies are found out with the help of service flow creation. In service flow creation message flow among services is found out.

After finding message flow among services communication among services is considered. In CBS implementation In a CBS implementation, the communication would most likely be handled by synchronous calls between the components. Synchronous communication means that the initiator is blocked from further computation until the requested component is providing the desiredanswer. The SOA approach however demands a loose coupling that allows more flexibility and better distribution of the components. Therefore asynchronous channels are used in addition to the synchronous ones. Asynchronous channels provide different reliability degrees or instance they canguarantee that every message is received either exactly once (EO) or even exactly once inorder (EOIO). Bothtypes of channels have mechanisms to ensure that each sent message is received exactly once, thus preventing message loss and multiple receiving of the same message. While messages on an EOIO channel are received in the same order as they are sent, messages on the EO channel may overtake each other. On EOIO channels, message racing is usually prevented by re-sorting the messages at the receiver side.

3.6 **Service Composition:**Service Consumer composes the services in order to provide a new or complex service to final user. In 2001 IBM introduced WSFL (Web Service Flow Language) to represent workflow of composite services. Same time Microsoft presented XLANG an extension of WSDL. In 2003, IBM and Microsoft combined them into a new Language BPEL4W (Business Process Execution Language). In 2004 BPEL4W has changed for WS-BPEL.

3.7 **Quality of Service:**There are two types of requirements; functional and non-functional requirements. Functional requirements express what software is aimed at. Non-functional requirements describe its quality parameters. Developers mostly focus on functional requirements during the service development (service identification, service design, service implementation, service usage) and mostly ignore non-functional requirements.

Galster and Bucherer (2008) have proposed taxonomy for identifying and specifying non-functional requirements in service oriented development. Quality model in service-oriented development process has four stages: service design, service discovery, service composition, runtime. Non-functional requirements vary stage to stage. Non-functional requirements can be categorized as: process requirement, non-functional service requirements (Sadiq J. et al. 2011).

3.7.1 Process Requirement:Process requirements are placed on the service-oriented development process. This process consists of service design, service discovery, service composition, and runtime. Process requirements become important when the customer wishes to influence the development process (e.g., in governmental or federal contracts). Some process requirements are:

- Implementation requirements
- Composition Requirements
- Standard requirements
- Cost constraints
- Time constraints
- Documentation requirements

3.7.2 Non-Functional Service Requirement:Non Functional Service Requirements are placed directly ontheservice-oriented system or an individual service under development. It is important to mention that these non-functional requirements can be derived directly from user needs, compared to the previous two categories of non-functional requirements which require more domain and environmental analysis.

Non Functional Service Requirements are difficult to handle, due to the highly distributed nature of service-oriented applications:

3.7.2.1 Usability: Usability in service-oriented systems is impacted by the data granularity used by services and by normal usability operations.

3.7.2.2 Reliability: Reliability can be separated into message reliability and service reliability. Message reliability addresses reliability of communication channels of the network over which services are made available. It therefore impacts the reliability of the whole system. Service reliability refers to the correct operation of an individual service. The use of standards can ensure a reasonable level of reliability of a service.

3.7.2.3 Performance:One individual service can have acceptable performance but due to network delays, high modularity and low cohesion within the system, the overall performance might decrease.

3.7.2.4 Safety : Often, safety is a quality which does not relate to the service alone but to the whole system. Standards can support safety, but for service-oriented systems not many mature standards exist . The risks involved in safety usually result from problems with the functionality of a system.

3.7.2.5 Security: As with safety, not many mature standards exist that could support this property. Failure in security can cause risks which impact safety.

3.7.2.6 Interoperability: Interoperability is the most prominent benefit of service-oriented systems. Through the use of standards (e.g., WSDL) interoperability is supported from a technical point of view. To allow full interoperability, a service must also provide semantic interoperability.

3.7.2.7 Availability: Service level agreements (SLA) cover availability agreements. To improve availability within a system, dynamic service discovery and composition in case of a service fall-out would increase availability.

3.7.2.8 Adaptability: No standards exist to support adaptability of a service so it is up to theservice user and service provider to manage adaptability.

3.7.2.9 Modifiability: An individual service can be difficult to modify, but a service-oriented system directly supports modifiability. A service interface must be designed carefully because changes that impact service users might be difficult to identify if the service is provided form external sources.

After the composition of the services application is ready for use.

4. CONCLUSION

In this thesis we have focused on WHAT should be done for service-oriented development instead of HOW it should be done. Proposed model presents service-oriented development process in a hierarchically structure and well defined way so that they can be used as reusable services. Service based development is much faster than component based software development; it reduces cost and effort of the system development using legacy services. Enhanced model offers flexible, loosely coupled and reusable service logics. Proposed model utilizes external (third party) services to supply business process. Model manages changes (variations) in the business process.

5. REFERENCES

1. Crnkovic I. and M. Larsson (2001). **“**Component-Based Software Engineering – New Paradigm of Software Development**”.** *Proceedings of the MRTC Publication.*
2. “Security in a Web Services World: A Proposed Architecture and Roadmap”, IBM/Microsoft, http://msdn2.microsoft.com/en-us/library/ms977312.aspx, v1.0, 7, 2002.
3. Wang, Y., Stroulia (2003), E.: “Flexible interface matching for web-service discovery. In: Web Information Systems Engineering (WISE)”. *Proceedings of the Fourth Inter-national Conference on*, 147–156.
4. Tsai T.W. (2005). “Service-Oriented System Engineering: A New Paradigm”. *Proceedings of the IEEE International Workshop on Service-Oriented System Engineering*, 0-7695-2438-9/05, pp. 3-6.
5. Michael N.Huhns and Munindar P. Singh (2005). “Service-Oriented Computing: Key Concepts and Principles”. *Proceedings of the IEEE Journal Internet Computing*, 1089-7801/05, pp. 75-81.

6. Alan W. Brown, Simon K. Johnston, Grant Larsen, Jim Palistrant (2005). SOA Development Using the IBM Rational Software Development Platform: A Practical Guide.
7. Kim Y. and H. Yun (2006). "An Approach to Modeling Service-Oriented Development Process". *Proceedings of the IEEE International Conference on Services Computing,* 0-7695-2670-5, pp. 273-276.
8. Ramollari E., D. Dranidis, and A. J. H. Simons (2006). "A survey of service oriented development methodologies". *Proceedings of the second Young Researchers' Workshop on Service Oriented Computing, Leicester, U.K.,* pp. 1-6.
9. Kokash, N., van den Heuvel, W.J., D'Andrea, V. (2006). "Leveraging web services discovery with customizable hybrid matching". *Proceedings of the Fourth Springer Interna-tional Conference on Service Oriented Computing (ICSOC),* pp. 522–528.
10. A.W. Brown and S.K. Johnston (2006), "A Model-driven Development Approach to Creating Service-oriented Solutions",*Proceedings of ICSOC06, International Conference on Service Oriented Computing*, pp. 624-636.
11. Zhang Y., t. Yo, Raman K. and K.J.Lin (2006). "Strategies for Efficient Syntactical and Semantic Web Services Discovery and Composition". *Proceedings of the eight IEEE International Conference on Enterprise Computing, E-Commerce and E-Services (CEC/EEE'06)*, 0-7695-2511-3.
12. Brown W. A., S. Iyengar, S.K. Johnson (2006), "A Rational Approach to Model-Driven Development", *IBM Systems Journal*,pp. 463-480, Vol. 44,http://www.research.ibm.com/journal/sj/453/brown.pdf
13. Papazoglou M. P., P.Traverso, S. Dustdar and Leymann F. (2007). "Service-Oriented Computing: State of the Art and Research Challenges". *Proceedings of the IEEE Journal,* 10.1109/MC.2007.400, pp. 38-45.
14. Bai X., D. Xu, Dai G. (2007). "Dynamic Reconfiguration Testing of Service-Oriented Architecture", *Proceedings of the 31st IEEE Annual International Computer Software and Application Conference (COMPSAC),* 0-7695-2870-8/07.
15. Tsai T.W., Jin Z., P.Wang and Wu B. (2007). "Requirement Engineering in Service-Oriented System Engineering". *Proceedings of the IEEE Conference on e-Business Engineering*, 0-7695-3003-6/07.

16. Xu.X, Mo.T, Wang.Z (2007). “SMDA: A Service Model Driven Architecture”. *Processing in the 3rd International Conference on Interoperability for Enterprise Software and Applications*, pp. 28-30.
17. S. Stein, Katja B., and M. E. Kharbili. “Enabling Business Experts to Discover Web Services for Business Process Automation”. http://www.ids-scheer.com/soa/
18. Chang H. S. and S. D. Kim (2008). “A Variability Modeling Method for Adaptable Services in Service-Oriented Computing”. *Proceedings of the eleventh IEEE Conference on Software Product Line,* 0-7695-2888-0, pp. 261-268.
19. Lopez S. M., C. J. Acuna, E. C. Cuesta and Marcos E. (2008). “Defining Service-Oriented Software Architecture Models for a MDA-based Development Process at the PIM level”. *Proceedings Seventh Working IEEE/IFIP Conference on Software Architecture,* 0-7695-3092-3, pp. 309-312.
20. Arsanjani et al. (2008). “A Method for Developing Service-Oriented Solutions (SOMA)”. *Proceedings of the IBM System Journal*, 0018-8670, vol. 43.
21. Engels G. and M. Assmann (2008). “Service-Oriented Enterprise Architectures: Evolution of Concepts and Methods”.*Proceedings of the twelfth IEEE Enterprise Distributed Object Computing Conference,* 978-0-7695-3373-5, pp. xxxiv – xliii.
22. DijkmanR.M. and M. Dumas (2008). “Service-oriented Design: A Multi-viewpoint Approach”, *Proceedings of the CiteSeer International Journal of Cooperative Information Systems.*
23. M. Galster, E. Bucherer (2008), “A Taxonomy for Identifying and Specifying Non-functional Requirements in Service-oriented Development”, Congress on Services, IEEE Computer Society.
24. Moosavi S., M. A. Seyyedi and N. Moghadam (2009). “A Method for Service Oriented Design”, *Proceedings of the Sixth IEEE International Conference on Information Technology: New Generations,* 978-0-7695-3596-8, pp. 290 – 295.
25. Gurupur V. and M. M. Tanik (2009). “Abstract Software Design Framework: A Semantic Service Composition Approach”. *Proceedings of the IEEE Conference on SouthEastCon,*978-1-4244-3976-8, pp. 295-300.

26. Wieczorek S. and A. Stefanescu (2009). “Service Integration: A Soft Spot in the SOA Testing Stack”. *Proceedings of the fifth Central and Eastern European Software Engineering Conference in Russia (CEE-SECR),* 978-1-4244-5664-2, pp. 211-216.
27. Balfagih Z., M. F. Hassan (2009), “Quality Model for Web Services from Multi-Stakeholders' Perspective”, *Proceedings of theInternational Conference on Information Management and Engineering*.
28. Dan A. and N. Priya (2009). “Dependable Service-Oriented Computing”, Proceedings of the IEEE Internet Coputing Journals, 1089-7801, pp. 11-15.
29. D. Becker (2009). “Service component architecture (SCA) lets you invokecomponents from different technologies”,
30. http://www.ibm.com/developerworks/opensource/library/os-apache-tuscany-sca/index.html?ca=drs-.
31. Bai L., J. Wei (2009). “A Service-Oriented Business Process Modeling Methodology and Implementation”. *Proceedings of the IEEE International Conference on Interoperability for Enterprise Software and Applications,* 978-0-7695-3652-1.
32. Web Services Reliable Messaging (WS-Reliable-Messaging), Vers. 1.1. OASIS Consortiom. Online at: http://docs.oasis-open.org/ws-rx /wsrm/v1.1/wsrm.pdf
33. Gholami F. M., J. Habibi, F. Shams, and S. Khoshnevis (2010). “Criteria-Based evaluation framework for service-oriented methodologies”, *Proceedings twelfth IEEE International Conference on Computer Modeling and Simulation,* pp.122-130.
34. Yang. L, Xing. K, Lee. S (2010), “A new conceptual life cycle model for Result-Oriented Product-Service System development”. *Proceeding of the IEEE International Conference on Service Operations and Logistics and Informatics (SOLI)*.
35. Fahmideh M., M. Sharifi, Jamshidi P., F. Shams and Haghighi H. (2011). “Process Patterns for Service-Oriented Software”. *Proceedings of the fifth IEEE Conference on Research Challenges in Information Science (RCIS),*978-1-4244-8670-0, pp. 1-9.
36. Leite S. D., C. Mary, F. Rubira, Castor F.(2011). “Exception Handling for Service Component Architectures”. *Proceedings 13th IEEE International Conference on Computer Modeling and Simulation.*

37. Sadiq J., A Mohsin and F. Arif (2011). "Quantifying Non-functional Requirements in Service Oriented Development". *Proceedings of theIEEE International conference on Frontiers of Information Technology (FIT),* 978-0-7695-4625-4, pp. 224-229.

38. Wang H. Y., J.C.Liao and C.H.Tsai (2011). "Objective Concept for Evaluating Service-Oriented Architecture", *Proceedings eighth international conference on Fuzzy System and Knowledge Discovery (FSKD),* 978-1-61284-181-6.