Quantitative Evaluation of AO based SOA Application for University Automation using an Experimental testbed

DHIVYA S
PG Scholar
SSN college of Engineering
Chennai, India
dhivyacse2309@gmail.com

SENTHIL VELAN S
Associate Professor
SSN college of Engineering
Chennai, India
senthilvelan@ssn.edu.in

ABSTRACT
Service Oriented Architecture (SOA) is a framework for building information systems by composing web services to form a business work flow. By introducing Aspect Oriented Programming in the logic of web service, it is possible to improve the quality attributes like maintainability, reusability and evolution. The functionalities modeled through the web services fuses the implementation of core and cross-cutting concerns. Business Process Execution Language (BPEL) is used for the composition of web services and models both core business logic and crosscutting functionalities. This paper address the evaluation of AO based SOA application for an application by using an experimental TestBed. The application developed for testing purpose is named as University Automation System. The services needed to model the application are created and deployed in Axis2 framework. By introducing Aspect Oriented Programming in the web services, a case can be made that the process improves the quality properties of the SOA application. A TestBed is a platform for conducting experiments on large development projects and evaluation of concepts using measurements. The proposed metrics focuses tier on three different tiers of software namely, core business tier, interface tier and access tier.

General Terms
Web service, composition, AOP

Keywords
AOP, SOA, BPEL, TestBed.

1. INTRODUCTION

Service Oriented Architecture (SOA) [1] modeled as an architecture for designing software that provides distributed web services based solution reflecting the requirements of a web enabled business application. The goal of SOA is to make available services to either end user applications or to other services in the web. The SOA applications are implemented through the knowledge of web services. The general characteristics of web services are reusable business components; which are loosely coupled and also considered as the building blocks of an SOA application. Web services can be described, published and invoked over the Internet using XML-based standards namely WSDL (Web Service Description Language) which contains description about web service, SOAP (Simple Object Access Protocol) which allows
switch over of data between different applications and UDDI (Universal Description Discovery and Integration) is a registry of web service interfaces described by WSDL.

A typical SOA application development involves designing software components for reuse and converting realized software components as web services and for service consumptions in end user applications. In SOA architecture model, the functionality is combined and packaged as inter operable services around business practice. The architecture also defines the IT infrastructure of a business and allows different applications to switch over data with one another as they participate in the business practice. By using web services Business Process Execution Language (BPEL) is a used for the definition and execution of business processes. It allow the top down approach of SOA[2] through composition, coordination and orchestration of web services.

Aspect Oriented Software Development (AOSD)[3] is a relatively new methodology to efficiently encapsulate the functionalities of a software and consequently increasing the modularity of the software. The methodology is able to achieve this property by modeling and implementing various functionalities of software into core and cross-cutting concerns. Core concerns represent the base functionalities that
addresses the core businesses like customer and account of a banking application. Cross-cutting concerns are functionalities that are scattered and tangled across the core business logic. These include logging, security, concurrency and transaction management that usually

cuts across many other modules of the software. Aspect Oriented Programming (AOP)[3] is a programming methodology to encapsulate these cross-cutting features through well defined unique constructs.

TestBed is an environment containing hardware and software tools to experiment and evaluate key ideas, methods and implementation for the application. The TestBed[4],[5] environment is designed to determine the impact using Aspect
Oriented Software Development (AOSD) methodology for the
development of an AO based application.

2. EXISTING WORK

In the literature survey, TestBed environment to test the working of a single web service has been proposed and implemented. To extend there are also proposed framework for generating TestBed infrastructure for SOA applications.

- Tsai[6] proposed a XML-based Object Oriented framework to test the web services. They perform various analysis such as dependency, completeness and consistency of the web service based applications.

3. PROBLEM AND SCOPE

Most of the existing work do not address TestBed for AO based SOA application. There is a need of TestBed to measure the impact on those application. This research work was based on creating a AOSD TestBed environment[5],[9] and to measure the impact on design properties for an AO based SOA application.

4. SERVICE ORIENTED ARCHITECTURE (SOA)

Service Oriented Architecture is modeled as an architecture to enables a designer to efficiently compose network enabled services resulting in development of SOA based applications. The Service Oriented Architecture models web services into three basic components:

- The Service Provider
- The Service Requester
- The Service Registry

The Service provider will publish (or unpublish) the offered services through the registry. This will enable the service requester to find the available services by searching the descriptions available in the service registry. Once the requester locates the desired service, the client binds with the identified service that is proposed by the service provider and invokes the service.

5. BUSINESS PROCESS EXECUTION LANGUAGE (BPEL)

Business Process Execution Language[2] is a language used for composing services available in the registry. In a BPEL process, any party to which the process interacts is called a partner, they may be the clients that invoke an available individual or composed service. It can also be a case in which external services can use this in individual or in their composition. A link defined to that party is called as partner link, so that some client can invoke the service. The WS-BPEL process workflow might look similar to the process flow shown in Fig 1.

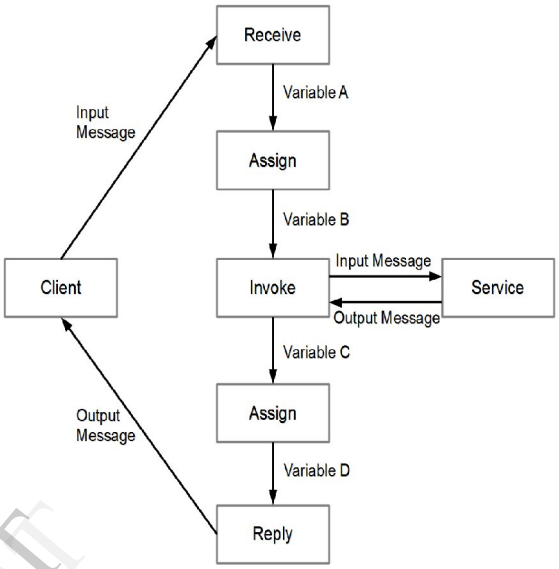

Fig. 1. Sample BPEL process

The basic steps involved in the working model of BPEL process is as follows,

1) Client invokes the BPEL process by sending a message.

2) The receive operation copies received data into variable A.

3) Assign operation crates variable B with data from variable A.

4) Invoke creates a service request from variable B.

5) Request sent to service by Invoke.

6) Response received by the Invoke.

7) Response message copied to variable C.

8) Assign creates variable D from variable C.

9) Reply creates service reply from variable D.

10) Service response is sent by the sub-process.

BPEL has two types of activities. They are primitive activities and structured activities. Some of the primitive activities includes invoke, receive, reply and assign. Some of the structured activities include sequence, flow, switch, while and pick.

6. ASPECT ORIENTED SOFTWARE DEVELOPMENT (AOSD)

Aspect Oriented Software Development is a new methodology, it aims to address the cross-cutting

concerns in software systems. Aspect Oriented Programming (AOP) [10] is a methodology that provides mechanisms for separation of cross-cutting concerns by introducing new constructs for encapsulating cross-cutting functionalities into modular units called, aspect. With AOP[3], we can model cross-cutting concerns as aspects instead of fusing them with the core modules.

An Aspect Weaver compile the final system by combining the core and cross-cutting modules through a process called weaving. AOP addresses the ability to separate concerns at multi-dimensional level and consequently increasing the modularity of the software. There are three steps in AOP methodology.

1. Aspectual decomposition - Decompose the requirements to identify cross-cutting and core concerns.
2. Concern implementation - Implementing the core and cross-cutting concerns independently.
3. Aspectual recomposition - Specifying the recomposition rules by creating modularization units, or aspect.

The cross-cutting constructs in the AOP model consists of two categories Common cross-cutting constructs and Dynamic cross-cutting constructs. A few common constructs consisting of

- Joinpoint - the point where the non-primary concern
  cross-cuts the primary functions of the code.
- Pointcut - a program construct that identifies join points.
- Aspect - the unit of encapsulation for cross-cutting concerns similar to classes in OOP.

Aspect supports dynamic cross-cutting constructs named as an advice. Advice are routines that are executed when a joinpoint is reached in the base code. It can be executed before, after or around the joinpoint.

7. ASPECT AWARE WEB SERVICE

We introduced aspect in web service to overcome the problems of tangling and scattering of concerns of a software. The web services need to be created in Axis2[11] Framework. Axis2 handles SOAP processing. Some of the identified requirements of Axis2 are

- Provide a framework to process the SOAP message.
- Ability to deploy a web service.
- Provide a client API to invoke web service.

The Axis war file is placed in the server. Axis2 is a service engine for deploying the web services. The Aspect for web service[12] need to be created separately. The Aspect Weaver, weave the aspect code into the web service at the run time is clearly depicted in Fig 2. Then the web service and aspect embedded web services are deployed by using Axis2. And they are composed by using BPEL Engine[13].

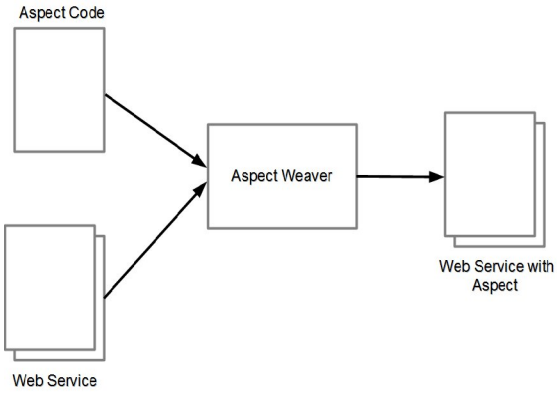

Fig. 2. Aspect Embedded Web Services

8. EXPERIMENTAL TESTBED

TestBed is an experimental environment designed using hardwares, softwares and tools to implement, execute, measure and infer on concepts related to technologies and methodologies. The SOA application services are created using the Axis framework[11], while some of the services are composed and deployed in the server. The AO based SOA application that was developed can be deployed in the experimental TestBed to infer on the implications of introducing AO in an SOA application. In this research on development of an AOSD TestBed environment[5], an experimental TestBed has been created to measure the impact on design level properties for a Aspect Oriented SOA application.

9. SYSTEM ARCHITECTURE

The introduction of Aspect in a web service plays a vital role in weaving of cross-cutting functionalities with the core functionalities. Once, the core web services and aspect embedded web services are created, they are composed to provide composite functionalities, ie, the core and aspect embedded web services are composed and interfaces are published to be used by a client. These services are then deployed so that the client can invoke the interfaces provided by the composed service[14]. Further, a TestBed environment is created to deploy the Aspect Oriented (AO) based SOA application, and to measure the impact on its design properties.

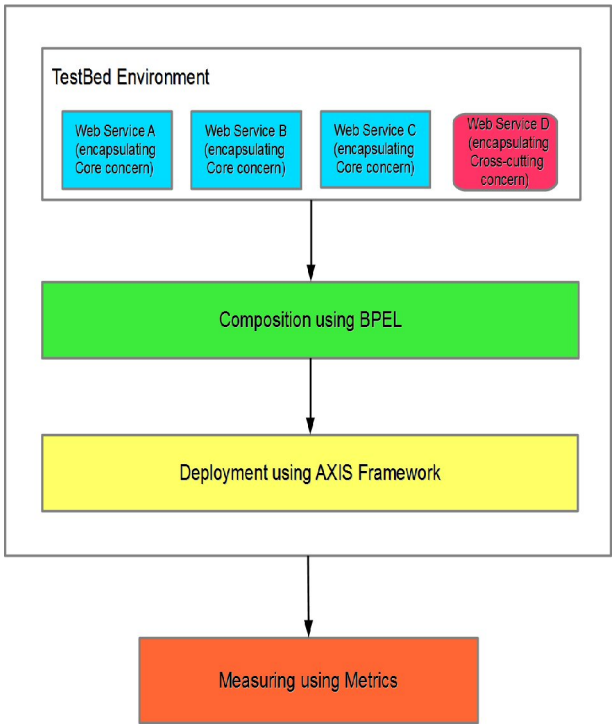

Fig. 3. AO Based TestBed Architecture

9.1. APPLICATION SCENARIO

The workflow for implementing this task, is clearly depicted in Fig 4. The student, who is going to register for the course will see the course from the course catalog. The schedule for the courses like timing, venue, and other necessary information can be obtained from the course schedule. The students can select and register for the course by specifying whether the course is counted as credit or only audit. The registered course can also be dropped until a date specified by the registration office. Students can also register courses without credit and considered as audit, to gain knowledge about the recent developments in a specific domain.

After registering for the course, classes are conducted, assignments are given to them, and finally they have to take the final exam. The exam pattern, mark distribution and other related activities depends upon the rules set by the faculty handling the course. The exam paper is evaluated by the faculty, and finally the result will be published on the web. If the result is success then the credits of the course are added, or else they need to re-appear for the exam.

The candidate admitted as a student needs to file a original plan of study one month from the date of starting the program. During the course the student can submit a revised plan of study. Students need to get required credits for completion of the course. The student has to apply for the degree and to be included in the graduates list for the next convocation.

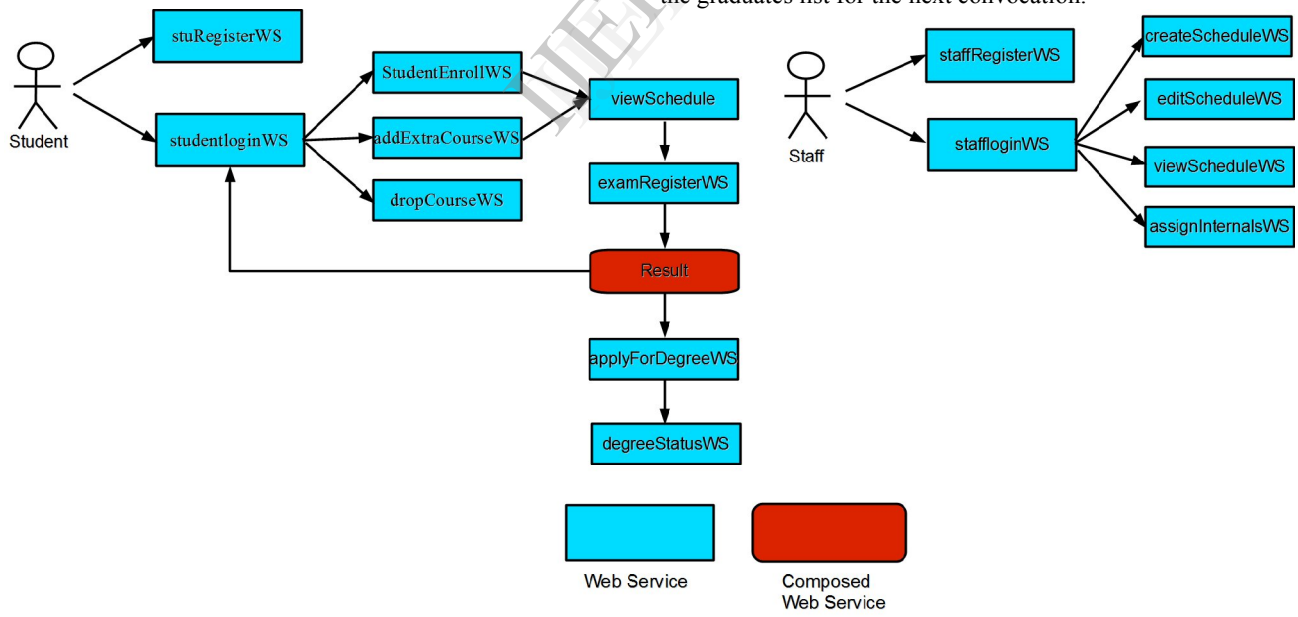

Fig. 4. Work Flow of University Automation System

The functionalities that are modeled as web services are listed below:

- createScheduleWS
- viewScheduleWS
- editScheduleWS
- studentEnrollWS
- addExtraCourseWS
- dropCourseWS
- examRegisterWS
- studentLoginWS
- staffLoginWS
- calInternalWS
- calSemMarkWS

- gradeCalculationWS
- displayResultWS
- stuRegWS
- staffRegWS
- applyForDegreeWS
- degreeStatusWS
- assignInternalsWS

9.2. ARCHITECTURAL VIEW OF UNIVERSITY AUTOMATION SYSTEM

Set of modules are identified in building the University Automation System are Web Service Creation, Web Service Composition, Web Service Deployment

1. Web Services Creation: The following are the ist of services that need to be created for University Automation System:
   - **i.** createScheduleWS- The schedule for the classes will be created by the faculty.
   - ii. viewScheduleWS- The Schedule can be viewed by both students and faculty.
   - iii. editScheduleWS- The Schedule can also be edited by faculty.
   - iv. studentEnrollWS- The Student should enroll for the courses offered during the next semester before the last date set by the University.
   - v. addExtraCourseWS- Any number of courses can be added by the students for each semester. They can add a ourse as either credit or audit.
   - vi. dropCourseWS- The course can also be dropped by the students during the progress of the semester.
   - vii. examRegisterWS- All registered students need to register in order to take the exam.
   - viii. studentLoginWS- A Student can login and register or drop the course.
   - ix. staffLoginWS- Each staff has a individual loginID.
   - x. CalInternalWS- Calculate total internal mark foe each course.
   - xi. calSemMark- Calculate the sum of internal mark and final exam mark and save the details in the table.
   - xii. GradeCalculationWS- Calculate grade for total marks and update in the table.
   - xiii. displayResultWS- Display the result for the requested register number.
   - xiv. stuRegWS- Student those who are newly admitted will register their details.
   - xv. staffRegWS- Staff those who are newly joined will register their details.
   - xvi. applyForDegreeWS- Check the condition whether the student completed all the courses and the total credits are reached.
   - xvii. degreeStatusWS- Calculate the cgpa earned and display the result.
   - xvii. assignInternalsWS- Staff register the internal marks earned by the students.

Some of the non-functional requirements in the University Automation System are

i) Logging - It contains the details about the service name, date and time of invocation of studentLoginWS, staffLoginWS services.

ii) Persistence - permanently stores student register number, course name, final exam mark and grade of gradeCalWS of the University Automation System.

The above listed non-functional cross-cutting concerns are encapsulated as aspects in AOP. The Aspect weaver will inject the aspects into the web services.

2. Web Services composition: BPEL[15] is used for the composition of web service and aspect embedded web services.

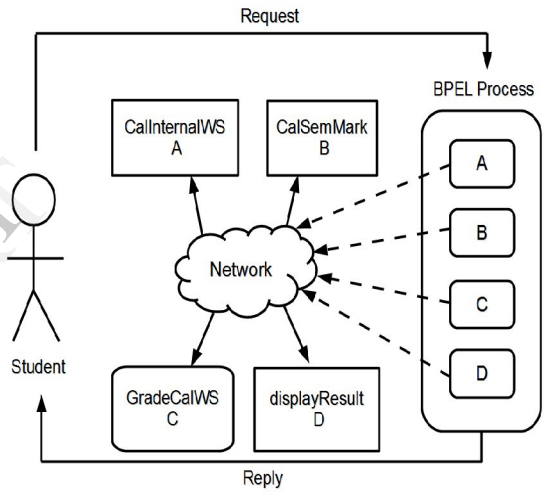

Fig. 5. BPEL Composition of web and aspect embedded web services

The services involved in composition are calInternalWS, calSemMarkWS, gradeCalculationWS and displayResultWS. The student will type the register number for seeing the result.

In the background process four services are composed and that will display the final result.

3. Deployment of Service: Web service and aspect embedded web services.[12] that are created must be deployed. Aspects are included to provide runtime process change, it is taken as the functionality where it is injected to the base process without altering the existing one. It can be (un)plugged at any point of time without touching or modifying the base processes.

10. PROPOSED MEASURES

Any proposed concept need to be evaluated to understand its strength and weakness. In order to do so an environment is necessary to implement and test the

concept. Also a good mechanism is necessary to quantitatively evaluate the concept using metrics.

Considering the proposed concept of AO TestBed, an environment has been proposed with a clearly defined architecture. In order to test its impact a set of metrics need to be clearly defined and evaluated. The set of metrics will focus on three parts, ie., core business metrics, interface metrics and access metrics.

While considering the core businesses, all the functionalities of the University Automation System need to be measured. And while calculating the quality of the interface the efficiency, usability, etc of the interface need to be quantified. Considering the access to permanent data storage the properties affecting the storage and retrieval of data from the database need to be evaluated. The non-functional requirements like logging and persistence, that are implemented using aspects need to be measured using well defined metrics.

11. CONCLUSION AND FUTURE DIRECTION

In order to measure the impact of using a methodology to develop an application software a good environment is essentially needed for testing. In this project an attempt has been made to develop and deploy base and aspect enabled web services for the automation of the processes in an University. The aspect enabled web services were used to encapsulate the cross-cutting functionalities of the application. The services are deployed in machines loaded with Windows7 and Ubunto Operating Systems. The deployed services are then composed by the BPEL engine using the composition rule specified in the BPEL configuration file. University Automation System was successfully tested for its execution. Three different parts of the application were identified for the definition of metrics. The future work is to measure the impact on design properties of University Automation System.

12. REFERENCES

[1] T. Erl, Service-Oriented Architecture: Concepts, Technology,and Design. Upper Saddle River, NJ: Prentice Hall Professional Technical Reference, 2005.

[2] Y. Vasiliev, Soa and Ws-Bpel, ser. From technologies to solutions. Packt, 2007. [Online]. Available: http://books.google.co.in/books?id=quRbYxaQ8wcC.

[3] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J. marc Loingtier, and J. Irwin, "Aspect oriented programming," in European Conference on Object Oriented Programming. SpringerVerlag, 1997.

[4] "Tao: A testbed for aspect oriented software development," http://www.comp.lancs.ac.uk/greenwop/tao/.

[5] P. Greenwood, A. Garcia, A. Rashid, E. Figueiredo, C. Sant'Anna, N. Cacho, A. Sampaio, S. Soares, P. Borba, M. Dosea, R. Ramos, U. Kulesza, T. Bartolomei, M. Pinto, L. Fuentes, N. Gamez, A. Moreira, J. Araujo, T. Batista, A. Medeiros, F. Dantas, L. Fernandes, J. Wloka, C. Chavez, R. France, and I. Brito, "On the contributions of an end-to-end aosd testbed," in Proceedings of the Early Aspects at ICSE: Workshops in Aspect-Oriented Requirements Engineering and Architecture Design, ser. EARLYASPECTS '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 8–. [Online]. Available: http://dx.doi.org/10.1109/EARLYASPECTS.2007.8

[6] W. Tsai, R. Paul, W. Song, and Z. Cao, "Coyote: an xml-based framework for web services testing," in High Assurance Systems Engineering, 2002. Proceedings. 7th IEEE International Symposium on, pp. 173–174.

[7] H. Holanda, G. Barroso, and A. Serra, "Spews: A framework for the performance analysis of web services orchestrated with bpel4ws," in Internet and Web Applications and Services, 2009. ICIW '09. Fourth International Conference on, May, pp. 363–369.

[8] L. Juszczyk and S. Dustdar, "Script-based generation of dynamic testbeds for soa," in in ICWS. IEEE Computer Society, 2010.

[9] P. Greenwood, A. Garcia, T. Bartolomei, S. Soares, P. Borba, and A. Rashid, "On the design of an end-to-end aosd testbed for software stability," in In Proceedings of the 1st International Workshop on Assessment of Aspect-Oriented Technologies (ASAT.07), 2007.

[10] R. Laddad, AspectJ in Action: Enterprise AOP With Spring, ser. Manning Pubs Co Series. Manning, 2009. [Online]. Available: http://books.google.co.in/books?id=ZmniOgACAAJ

[11] "Apache axis2 architecture guide," http://axis.apache.org/axis2/Axis2ArchitecturGuide.html.

[12] N. C. Mendonca and C. F. Silva, "Aspectual services: Unifying service- and aspect-oriented software development," in Proceedings of the International Conference on Next Generation Web Services Practices, ser. NWESP '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 351–. [Online]. Available: http://dx.doi.org/10.1109/NWESP.2005.20

[13] A. Charfi and M. Mezini, "Ao4bpel: An aspectoriented extension to bpel," World Wide Web, vol. 10, no. 3, pp. 309–344, Sep. 2007. [Online].Available: http://dx.doi.org/10.1007/s11280-006-0016-3

[14] J. Zhang, F. Meng, and G. Liu, "Research on soa-based applications based on aop and web services," in Proceedings of the 2008 International

Conference on Computer and Electrical Engineering, ser. ICCEE '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 753–757. [Online]. Available: http://dx.doi.org/10.1109/ICCEE.2008.107

[15] A. Charfi, Aspect-oriented Workflow Languages: AO4BPEL and Applications, 2007. [Online].Available: http://books.google.co.in/books?id=YV2mNwAACAAJ

S.Dhivya is currently pursuing M.E in Computer Science and Engineering at SSN College of Engineering. She has completed B.Tech in Computer Science and Engineering at Kalasalingam University.

S. Senthil Velan is currently an Associate Professor in the Department of Computer Science and Engineering at SSN College of Engineering. He has completed B.E in Electronics and Communication Engineering at Madurai Kamaraj University and M.S in Computer Science at Wichita State University. He is currently pursuing the Ph.D program at Anna University in the field of Aspect Oriented Software Development. The author has a teaching experience of around 15 years and 3 years of industrial experience.