PROTOTYPE SERVICE ORIENTED ARCHITECTURE BASED REMOTE EXPERIMENTATION FOR HIGH POWER RF SYSTEM AT 3.7GHz

By

Mitesh Kumar Sharma
(05MCE006)

Guide

Dr. Promod Sharma
Mr. Ritesh Sugandhi

A Dissertation
Submitted to

Nirma University of Science and Technology
in partial fulfillment of the requirement
for the degree of

Master of Technology

Department of Computer Science & Engineering
Ahmedabad 382481
May 2007

This is to certify that Dissertation entitled

PROTOTYPE SERVICE ORIENTED ARCHITECTURE BASED REMOTE EXPERIMENTATION FOR HIGH POWER RF SYSTEM AT 3.7GHz

Submitted by

Mitesh Kumar Sharma

has been accepted toward fulfillment of the requirement
for the degree of
Master of technology in Computer Science & Engineering

Prof. S. N. Pradhan
Professor In Charge

Prof. D. J. Patel
Head of The Department

Prof. A. B. Patel
Director, Institute of Technology

ACKNOWLEDGEMENT

This indeed is the moment of proud pleasure to present "Prototype Service Oriented Architecture Based Remote Experimentation for High Power RF System at 3.7 GHz" - a work of comparatively shorter duration that would go a long way in making the systems.

Starting off, I am indebted to **Dr. S.N. Pradhan** (M.Tech. Coordinator, Computer Science & Engineering Department, Nirma University, Ahmedabad) because of whose sincere efforts I managed to get opportunity to work for an evolving project at **Institute for Plasma Research** (IPR), a prestigious research institute.

Getting an opportunity to work in one of the country's most scientifically organized institution, is like a dream come true. My limited association with IPR has provided me an unparallel exposure to the challenging task of developing the system, which is the first ever of its type, in a very short period of time. Exposure to the work culture, discipline, high professionalism and dedication, I was so fortunate to get while working at IPR, has immensely benefited my perspective and has left deep imprints on my mind to last throughout my career.

I do not know how to express my gratitude to the Institute, the authorities of IPR and more particularly to my project supervisor **Dr. Promod Sharma** and project guide **Mr. Ritesh Sugandhi**, sufficiently for their constant guidance, encouragement and moral support which helped me to accomplish the project.

I am thankful to all faculty members of Department of Computer Science & Engineering, Nirma University, Ahmedabad for their special attention and suggestions towards the project work.

The blessings of God and my family members makes the way for completion of major project. I am very much grateful to them.

Mitesh Kumar Sharma
Roll No.05MCE006

ABSTRACT

Lower Hybrid current Drive system (LHCD) is a vital system in Steady State Super Conducting Tokamak -1 (SST-1). It would be used for non inductive current drive and heating Experiments. LHCD System houses 4 klystron based high power RF system at 3.7 GHz. Each High power RF system operation is supplemented by the boot up and control of auxiliary power supplies and microwave system. For the long term research objective and collaboration with international/national group, it is desirable to setup a mechanism for the online remote experimentation and data sharing across the web. The distributed and heterogeneous instrumentation and control propose the need of application integration. The recent development in Service Oriented Architecture (SOA) and Web Services promises a great deal in solving similar issues. Hence, a SOA based remote experimentation system has been indigenously designed, developed and tested. The data archival and visualization software has been developed in the MDS+ data acquisition system which is widely used in tokamak research and development community. The performance issues were considered to be of importance and a set of studies has been carried out. The experiments with Simple Object Access Protocol (SOAP) messages have been made in a Web Service Environment to find out the performance in response time using SOAP. The naive and direct use of SOAP resulted in higher response time but with the improvements and an intelligent choice of eXtended Markup Language (XML) parser, the response time for SOAP messaging have been improved drastically.

The functional prototype of the implementation has been tested with the VersaModule Eurocard (VME) based Data Acquisition and Control Setup at LHCD Lab. Finally, this product will be a part of SST-1 LHCD system data acquisition and control activities.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS & ABBREVIATIONS

LHCD	Low Hybrid Current Drive
SOAP	Simple Object Access Protocol
SST-1	Steady State Super Conducting Tokamak -1
SOA	Service Oriented Architecture
VME	Versa Module Eurocard
DAC	Data Acquisition and Control
UML	Unified Modeling Language
PS	Power Supply
WS	Web Service
QoS	Quality of Service
W3C	World Wide Web Consortium
XML	eXtended Markup Language
HTTP	Hyper Text Transport Protocol
WSDL	Web Service Description Language
UDDI	Universal Description, Discovery and Integration
DOM	Document Object Model
SAX	Simple API for XML
TCP/IP	Transmission Control Protocol/Internet Protocol

1. INTRODUCTION

1.1 INTRODUCTION TO INSTITUTE FOR PLASMA RESEARCH

In 1981-1982 the Department of Science and Technology (DST), Government of India recognized plasma physics program (PPP) which was initiated as an independent activity under the shadow of PRL. Institute for Plasma Research (IPR) was established in 1986, as an autonomous institution, funded by the Department of Atomic Energy. The major objectives of institute are to carry out fundamental theoretical and experimental research on magnetically confined hot plasmas and nonlinear plasma phenomena. The institute has various working groups, like Pulsed Power Group, RF Group, Cryogenics Group, Vacuum Group, Modeling Group, Neutral Beam Injection Group, etc. The Institute for Plasma Research is a multidisciplinary center of excellence for basic and applied research in charged particle beams, plasmas, high power microwave generation, and millimeter-wave electronics, advanced high energy particle accelerators, ion beam micro fabrication, nonlinear dynamics (chaos) and more. Much of the center's research involves plasma, an ionized gas that makes up some 98 of the visible matter in the universe. Plasma plays an important role in Controlled Thermonuclear Fusion (CTF); both in magnetically confined plasma and in plasma created and compressed by lasers or heavy ion beams. CTF reactors could provide a lasting solution to the world's energy problem, so this area of study is one focus for research at IPR.

1.2 STEADY STATE SUPER CONDUCTING TOKAMAK -1 PROJECT

The word tokamak means "toroidal chamber" in Russian. It is a magnetic fusion device that is in a shape of a torus as shown in Figure 1.1, and which depends on external windings for generating a strong toroidal magnetic field. Poloidal magnetic fields are created primarily by a toroidal current inside the plasma itself and then are processed. This combination of toroidal and poloidal magnetic fields generate an overall nested helical structure which is necessary to keep the plasma stable. The tokamak is presently the leading candidate design for a future "working" magnetic fusion device.

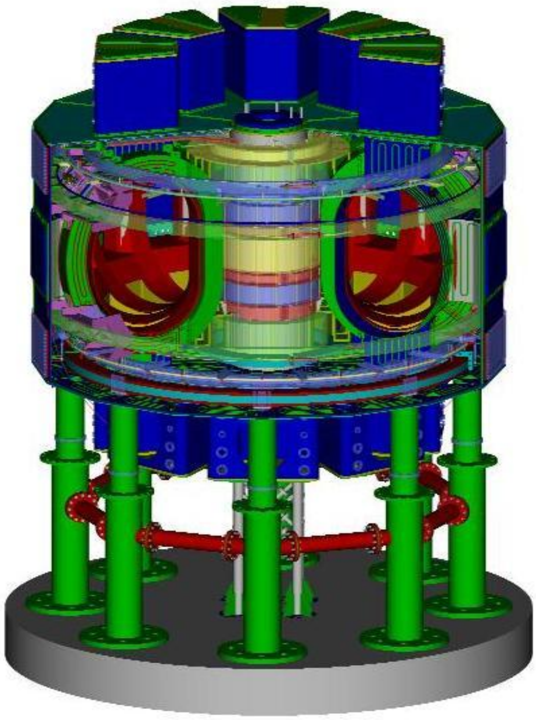


Fig 1.1: 3D View of SST-1 Tokomak

A steady state super conducting Tokamak SST-1 is a prestigious national project at IPR. The objectives of SST-1 include studying the physics of the plasma processes in tokamak under steady state conditions and learning technologies related to the steady state operation of the tokamak. These studies are expected to contribute to the tokamak physics database for very long pulse operations. The specific objective of the SST-1 project is to produce 1000 s elongated double null diverter plasma. There are several conventional questions in tokamak physics, which will be addressed again in steady state scenario. Some of these are related to the energy, particle and impurity confinement, the effect of impurities and edge localized modes (ELM) in steady on energy confinement, the stability limits and their dependence on current drive methods, the resistive tearing activities in presence of RF fields, disruptions and vertical displacement events (VDE), and thermal instability. In steady state operations non-inductive current drive will sustain the plasma current. Different aspects of current drive such as different current drive methods and their combinations, current drive efficiency, profile control and bootstrap current, are studied at IPR.

1.3 LOWER HYBRID CURRENT DRIVE SYSTEM RESEARCH AT IPR

A Lower Hybrid Current Drive (LHCD) system would be the prime method to drive current in SST-1. A typical layout of the LHCD system is shown in Figure 1.2 [1]. The LHCD system is designed to couple 1 MW of RF power at 3.7 GHz to SST-1 Tokamak. The High power RF components technology development and Lower Hybrid current drive research in various plasma condition forms the core research activities of the group. The LHCD system is centered on a Klystron Amplifier based RF sources for power generation. The generated power would be transfer to SST-1 tokamak through 64 WR-284 arms in vessel system. The klystron amplifier base high power RF system is procured to deliver 500 KW of RF power at 3.7 GHz. The Klystron Amplified electrode is powered by a set of indigenously developed high voltage DC power supplies. High power RF system Commissioning, high power RF Component development, VME based automated control system operation for RF source testing and Power Coupling experiment in ADITYA Tokamak are the major achievement of the group. [2]

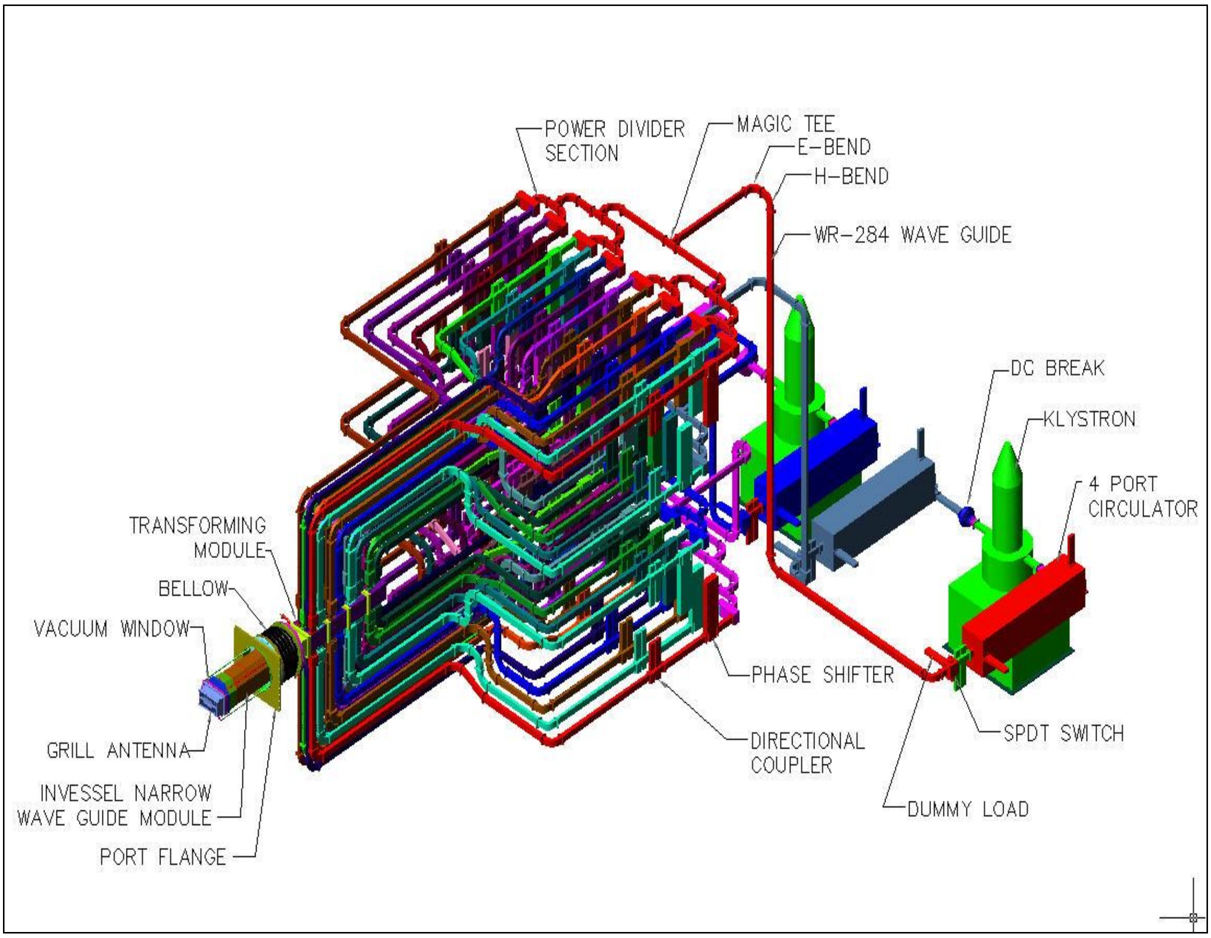


Fig 1.2: Typical Layout of LHCD System

1.4 HIGH POWER RF SYSTEM

The Klystron based high power (500KW RF) source is commissioned by LHCD group for SST-1 project. Centered on it, a set of indigenously developed power supplies and Data acquisition and Control System is developed in house. This system finds its usefulness in understanding High power RF operation on ADITYA Tokamak and indigenously developed High power Components e.g. Dummy load. The system is one of its kinds in India. Finally this system would be a part of SST-1 LHCD system. . The system block diagram and the snapshot of the system are shown in Figure 1.3 and 1.4 respectively.

1.4.1 Components of High Power Test System

The high power RF system compromised of a set of subsystems as shown in Figure 1.3 which are described as below: [1]

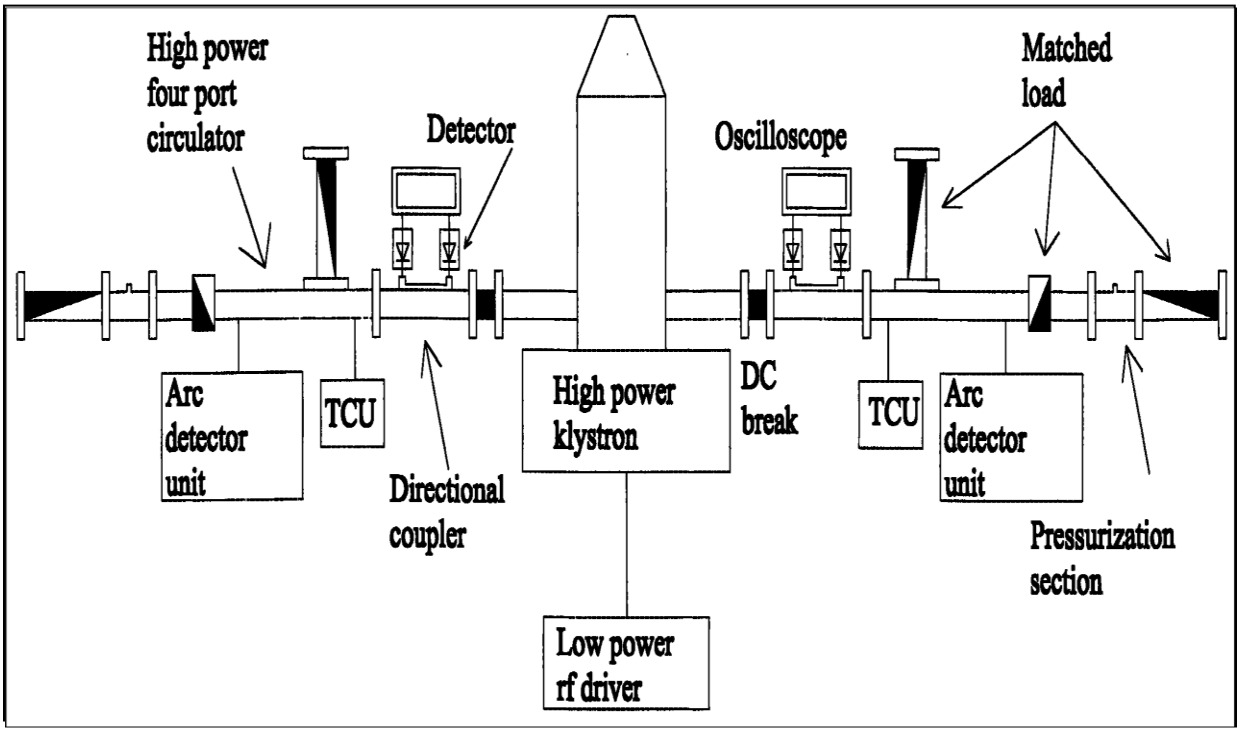


Fig 1.3: The Klystron Setup showing Schematic of Various RF Components.

1.4.1.1 Klystron RF source

The klystron is a RF amplifier. The Klystron Device (2103) used in high power system is procured from Thomson CSF. The main characteristics (nominal) of the tube are shown in Table 1.1.

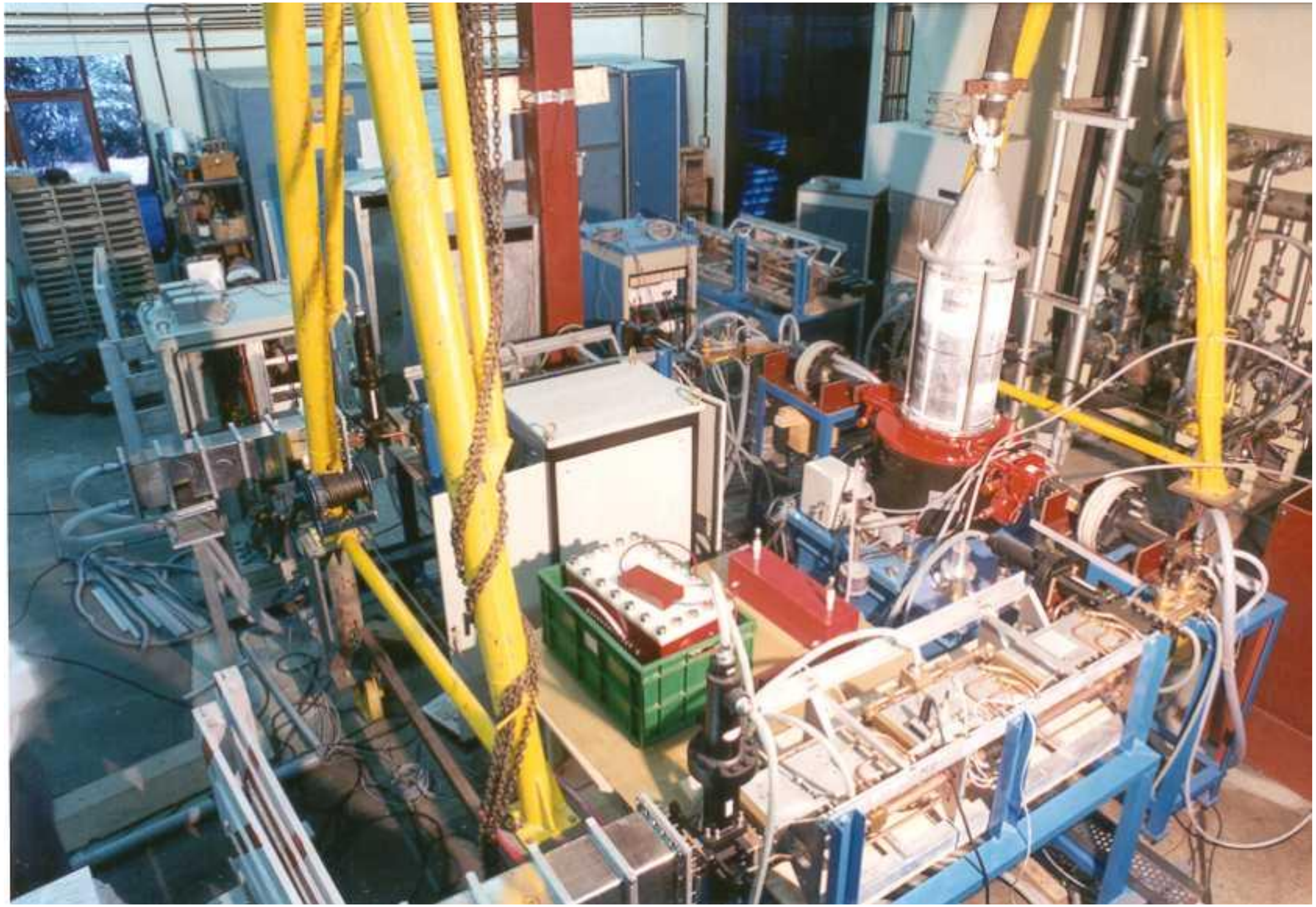


Fig. 1.4: Actual Klystron Setup showing Various Electrical Components.

Table 1.1: Characteristic Parameter of Klystron Tube

Frequency (MHz)	3700 ±10
Nominal input power (W)	~7
Nominal output power (kW)	500 KW (1000 s)
Gain (dB)	~50
Efficiency (%)	~45
Cathode voltage (kV)	~65
Beam current (A)	~20
Mode	CW

1.4.1.2 Lower Power Amplifier Section

The LPS is a 1-7 watt RF power source used to feed the input to the klystron. The klystron amplifiers amplify this power and give the output power of 0-250 KW in two arms. The LPA is a phase locked RF source. The LPA is operated through DAC by setting the input voltage and pulse determines the operation time. The operation of LPA is synchronous with a pulse with characteristics parameters as Ontime, Offtime, StartVoltage, DeepVoltage etc. Typically 7 W of input RF power is fed into the first cavity of the klystron from a low power RF source through an N-type coaxial connector. A phase locked oscillator is used to generate low RF power.

1.4.1.3 Ion Pump Power Supply 1 and power supply 2

This power supply is connected to the ion pump of klystron tube to create and maintain the required vacuum into the klystron tube arm 1 and arm 2 respectively. These are remotely operated through the DAC and interfaces with the DAC through following signals as shown in Table 1.2.

Table 1.2: Signal I/O Requirement for Ion Pump Power Supply

S.N.	Signal Name	Type
1	Ion Pump power supply Start	Digital Output
2	Ion Pump Power Supply Stop	Digital Output
3	Ion Pump Power supply Current	Analog Input
4	Ion Pump Power Supply ON	Digital Input
5	Ion Pump Power Supply Ready	Digital Input
6	Ion Pump Power Supply Trip	Digital Input
7.	Ion Pump Power Supply Current Set	Analog Output

1.4.1.4 Magnet Power Supply 1

The electron beam is focused through different cavities using magnets. The magnet power supply 1 is used to power the beam focusing magnet 1 of the klystron tube. The magnet power supply 1 interfaces with the DAC through following signals as shown in Table 1.3.

Table 1.3: Signal I/O Requirement for Magnet Power Supply 1

S.N.	Signal Name	Type
1	Magnet Power Supply 1 Start	Digital Output
2	Magnet Power Supply 1 Stop	Digital Output
3	Magnet Power Supply 1 Current	Analog Input
4	Magnet Power Supply 1 Voltage	Analog Input
5	Magnet Power Supply 1 ON	Digital Input
6	Magnet Power Supply 1 Ready	Digital Input
7	Magnet Power Supply 1 Trip	Digital Input
8	Magnet Power Supply 1 Current Set	Analog Output

1.4.1.5 Magnet Power Supply 2

The magnet power supply 2 is used to power the beam focusing magnet 2 of the klystron tube. The magnet power supply 2 interfaces with the DAC through following signals as shown in Table 1.4.

Table 1.4: Signal I/O Requirement for Magnet Power Supply 2

S.N.	Signal Name	Type
1	Magnet Power Supply 2 Start	Digital Output
2	Magnet Power Supply 2 Stop	Digital Output
3	Magnet Power Supply 2 Current	Analog Input
4	Magnet Power Supply 2 Voltage	Analog Input
5	Magnet Power Supply 2 ON	Digital Input
6	Magnet Power Supply 2 Ready	Digital Input
7	Magnet Power Supply 2 Trip	Digital Input
8	Magnet Power Supply 2 Current Set	Analog Output

1.4.1.6 Magnet Power Supply 3

The magnet power supply 3 is used to power the beam focusing magnet 3 of the klystron tube. The magnet power supply 3 interfaces with the DAC through following signals as shown in Table 1.5.

Table 1.5: Signal I/O Requirement for Magnet Power Supply 3

S.N.	Signal Name	Type
1	Magnet Power Supply 3 Start	Digital Output
2	Magnet Power Supply 3 Stop	Digital Output
3	Magnet Power Supply 3 Current	Analog Input
4	Magnet Power Supply 3 Voltage	Analog Input
5	Magnet Power Supply 3 ON	Digital Input
6	Magnet Power Supply 3 Ready	Digital Input
7	Magnet Power Supply 3 Trip	Digital Input
8	Magnet Power Supply 3 Current Set	Analog Output

1.4.1.7 Filament Power Supply

This power supply is used to power the filament of the klystron tube which in turns emits the electrons toward klystron collector. The filament power supply interfaces with the DAC through following signals as shown in Table 1.6.

Table 1.6: Signal I/O Requirement for Filament Power Supply

S.N.	Signal Name	Type
1	Filament Power Supply 3 Start	Digital Output
2	Filament Power Supply 3 Stop	Digital Output
3	Filament Power Supply 3 Current	Analog Input
4	Filament Power Supply 3 Voltage	Analog Input
5	Filament Power Supply 3 ON	Digital Input
6	Filament Power Supply 3 Ready	Digital Input
7	Filament Power Supply 3 Trip	Digital Input
8	Filament Power Supply Voltage set	Analog Output

1.4.1.8 Anode Modulator Power Supply

The anode modulator power supply is used to power the anode of the klystron amplifier which in turn accelerates the electron beam inside the klystron tube. The anode modulator power supply interfaces with the DAC through following signals shown in Table 1.7.

Table 1.7: Signal I/O Requirement for Anode Modulator Power Supply

S.N.	Signal Name	Type
1	Anode Modulator Power Supply Start	Digital Output
2	Anode Modulator Power Supply Stop	Digital Output
3	Anode Modulator Power Supply Current	Analog Input
4	Anode Modulator Power Supply Voltage	Analog Input
5	Anode Modulator Power Supply ON	Digital Input
6	Anode Modulator Power Supply Ready	Digital Input
7	Anode Modulator Power Supply Trip	Digital Input
8	Anode Modulator Power Supply Voltage set	Analog Output

1.4.1.9 High Voltage Power Supply

The high voltage power supply is used to power the cathode of the klystron tube. The voltage operation of the power supply is through an electromechanical device operated by the pulse. The pulse length is used to determine the voltage to be achieved between 0-70 KV. The high voltage power supply interfaces with the DAC through following signals shown in Table 1.8.

Table 1.8: Signal I/O Requirement for High Voltage Power Supply

S.N.	Signal Name	Type
1	Vacuum Contact Breaker (VCB) Start	Digital Output
2	Vacuum Contact Breaker (VCB) Stop	Digital Output
3	Beam Current	Analog Input
4	Beam Voltage	Analog Input
5	Vacuum Contact Breaker (VCB) ON	Digital Input
6	Vacuum Contact Breaker (VCB) Ready	Digital Input
7	Vacuum Contact Breaker (VCB) Trip	Digital Input
8	Voltage Raise Pulse	Digital Output
9	Voltage Lower Pulse	Digital Output

1.4.1.10 Klystron High Power Output section

Klystron has two output wave guide arms which are connected by a switch to transmission line or dummy load. Sets of measuring devices are connected which give the information about the output power and arc detection in the tube.

The RF power is taken out from the klystron by means of two rectangular WR-284 waveguides. The efficiency of the klystron is typically 45% and therefore a large amount of cooling is required to remove the dissipated beam power at the collector. For this purpose, demineralised and deionised (DMDI) water is passed through the collector. The RF power from the output WR-284 waveguides is fed into the water cooled high power dual directional couplers to monitor the forward and reflected power using coaxial microwave detectors. The output from the directional coupler is connected to the input of the four-port high power circulators to protect the klystron from high reflection. An arc detector unit is installed near the circulator mouth to detect any arcing in the klystron as well as waveguides. As soon as the arc is detected by the arc detector, the applied high voltage is removed by tripping a vacuum circuit breaker (VCB) and simultaneously triggering the rail gap switch. The output arm of the circulator is connected to the dummy load which is water cooled. The klystron high power output interfaces with the DAC through following signals shown in Table 1.9.

Table 1.9: Signal I/O Requirement for Klystron High Power Output Section

S.N.	Signal Name	Type
1	Forward Power	Analog Input
2	Reflected Power	Analog Input
3	Arc Detector 1	Digital Input
4	Arc Detector 2	Digital Input

The all above systems are nonlinear RF devices. Hence, calibrations and interpolations are required for configuring and controlling these devices.

1.5 LHCD DATA ACQUISITION AND CONTROL SYSTEM

The LHCD system on SST-1 is aimed for the non-inductive current drive and heating. The Data Acquisition and Control System (DAC) is a vital subsystem. The objectives of the LHCD DAC System can be described as below:

1.5.1 Core Objectives

The prime objective of LH-DAC is to provide the facility for remote operation of LHCD system. It helps the user to plan the experiment systemically. It safely interacts with the instruments and control systems and configures system settings as per operational requirements defined prior to the operation. It provides protected system operation through two mechanisms, one is hardwired approach which takes less than 10 microseconds and another one is software interlocked system operations for critical systems. LH-DAC provides real time data acquisition for less than 1 millisecond and generates pulses during system operation. It provides interactive Graphical User Interface (GUI) to the user so that the controlling and handling of various complicated operations can be performed easily. Besides this, it provides the facility for storage of signals in database so that it can be retrieved later for analytical and research purpose. In this way, it maintains the system operation logbook. It communicates and exchanges data between SST-1 control system and LHCD system. One of its main objectives is to provide web based data analysis and user management.

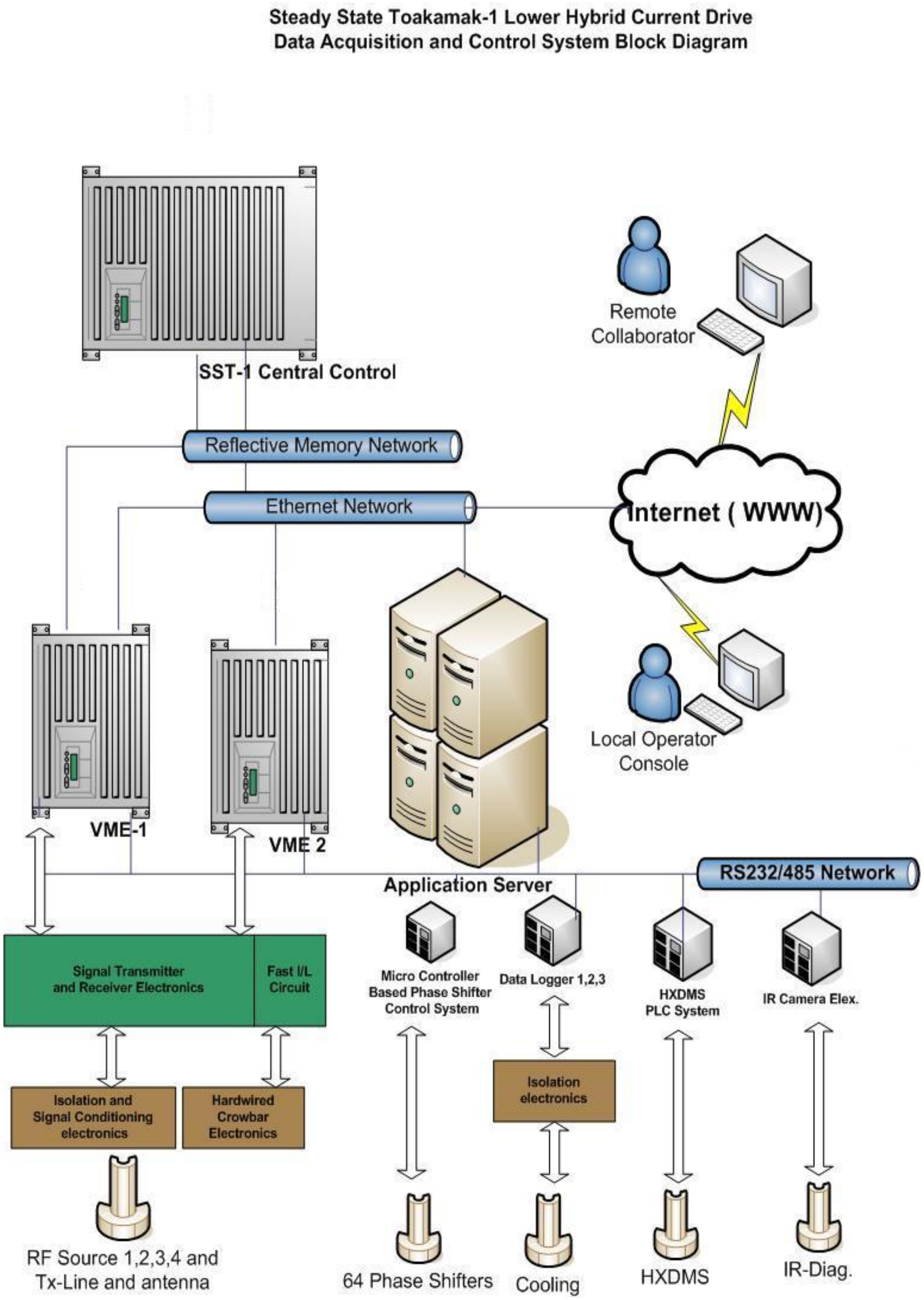


Fig 1.5: The SST-1 LHCD Acquisition and Control System Block Diagram

Steady State Tokamak -1 LHCD DAC system structure is shown in Figure 1.5. It consists of a number of components which are classified in 4 tiers. The architecture and the functioning of this system as shown in Figure 1.5 are described in detail in chapter 4.

1.5.2 Functional Decomposition

The system as shown in Figure 1.5 is broadly divided into following 4 tiers which are mentioned as follows:

1. **Human Machine Interface**
   This tier is responsible for the human machine interaction and verification of the request and presenting the raw data into suitable form.
2. **Application Software Tier**
   This tier is responsible for implementation of the control logic of the experiment and provides the system services to the users.
3. **Instrumentation and Control Tier**
   This tier provides the required signal instrumentation and control modules programming and control provider.
4. **Signal processing Electronics**
   This tier provides the signal conversion /processing/isolation electronics. The cables layout plan and connection plan is taken care by this layer.

The application software developed in the project forms a part of application software tier of the LHCD DAC activities. This software is used to control above discussed high power RF system. The detailed requirement descriptions are presented in chapter 2.

2. REQUIREMENT SPECIFICATION

This chapter discusses the objectives of the system. In it, the various requirements of the system are addressed. The requirements of the different users and the actions expected to be performed are described with the help of UML diagrams. The requirement specification sheet is also provided to understand the actions required.

2.1 REQUIREMENT CAPTURING

The requirement capturing and specification representation is the most important phase of software development life cycle. In this project, adequate attention is paid to this phase. The following core points highlight the requirements of the system are as follows.

1. Computing anywhere and anytime

One of the core objectives of the SST-1 is to learn the physics and technological challenges for steady State operation of the tokamak. The world is opening its wings. In this era, a number of national and international groups are going for collaborative research so that costly experimental facilities can be used for the studying various physics issues. Under national fusion program, IPR is also associated in such activities. Hence, it is desirable to setup such mechanism to operate experiment from any part of the world. This will bring the functional flexibility to the operations and ideas sharing.

2. Integrated remote operation of High power RF system

As mentioned in the chapter one, High Power RF System comprised of various heterogeneous and distributed signal processing instruments and control hardware. Each hardware is controlled by a set of controller consists of software and libraries. The distributed operation becomes difficult to operate from user point of view. Hence, it makes a sense to have a single point of operation console.

3. Remote online monitoring of system behavior

The system behavior needs to be observed by the supervisor. The system is supposed to be operated in the interactive mode with online updating of the voltage and current waveforms. The online monitoring is of the order of ~250ms.

4. Authenticate and secure hierarchical access

The authentication and security make a sense to provide a protected operation of the system. It is required that the expert who is familiar with operating the system must operate otherwise causalities can lead to loss of system as well as of man power also.

5. Web base Archived Data and log book visualization

Web is easily available medium for data sharing and visualization. Hence looking at the remote participation objective of the project, there is a feeling to publish the results on the web so that it can help the scientist and researchers for analysis across the globe.

6. Remote enhancements of the application software

The subsystems are non-linear in nature and the system components need to be conditioned and recalibrated. Due to research environment the requirement are rather experiment oriented and the dynamic in nature. If system could provide the remote deployment and up gradation of the software the system designer can be consulted on web. The physical presence of the system designer can be minimized.

7. Data representation

The Data representation compatible to environment and users familiarity leads to assist in analysis and which in turn leads to Ideas sharing. A set of tokamaks has proposed and tried out various mechanism and tools. The MDS+ is widely used Data archival system with physical installation at more than 50 tokamaks world wide. It is based on fusion grid architecture and plasma physicists are familiar with the tools like jScope. Hence, it looks attractive and meaningful to go on the same path.

First, the requirements are written into simple English and later on, UML user case modeling approach has been followed to give proper shape to the requirement specification.

2.2 USERS IDENTIFICATION

The following set of users is supposed to interact with the system and they are identified as below:

1. **Experiment Planner:** A person having this role should be able to plan the project. It is one who can define the experimental plan and a set of test plans. In order to assist him the system can provide the theoretical simulators and logs.
2. **Experiment Executor:** The Experiment Executor is a person who can execute and supervise the experiment execution as per the plan.
3. **Experiment Viewer:** The experiment viewer is a person who can just see the experimental result and useful information.
4. **System Administrator:** The System Administrator is responsible for the software component administrating as well as regular data backup and maintenance tasks.

2.3 SCOPE AND BOUNDARY OF THE PROJECT

With reference to LHCD-DAC software diagram as shown in previous chapter (Figure 1.5), this project is supposed to develop the application level functionalities. The system interacts with the VME based system instrumentation and control system by TCP/IP based command execution. The application software is required to be platform transparent and must be made out of flexible functional components and modular in nature so that further enhancement or modification in one block does not affect the other one.

2.4 REQUIREMENTS AND USE CASE DIAGRAMS

The requirements of the project can be classified into three different modules according to the set of users identified above. The purpose of the modules, the expected interaction with the system and the parameters supplied by these modules are explained in detail and use case diagrams are associated to help in understanding the need.

Use case diagram is "a diagram that shows the relationships among actors and use cases within a system." Use case diagrams are often used to:

Provide an overview of all or part of the usage requirements for a system or organization in the form of an essential model or a business model

Communicate the scope of a development project

Model the analysis of usage requirements in the form of a system use case model

2.4.1 Experiment Planning Requirements

The experiment planning refers to configure parameters and setting test plans. Due to scientific nature of the project, it is a specialized activity. A use case diagram for the experiment planning requirements is shown in Figure 2.1. Planning includes the following purposes as below.

1. Declare the purpose of the experiment
    a. Experiment on ADITYA Tokamak
    b. Experiment on SST-1 Tokamak
    c. RF Conditioning
    d. Normal Conditioning
2. Declare the input parameters of the system
    a. Magnet Power Supply 1
    b. Magnet Power Supply 2
    c. Magnet Power Supply 3
    d. Filament Power Supply
    e. Anode Modulator Power Supply
    f. High Voltage Power supply
    g. Ion Pump Power Supply
    h. Input RF power
3. Publish the experimental data sheets for input parameters
4. Set the test cases for the planned experiment

    The test case refers operational regime of the klystron based RF system.
5. Save and record the experiment

Post Condition:

The experiment plan is defined.

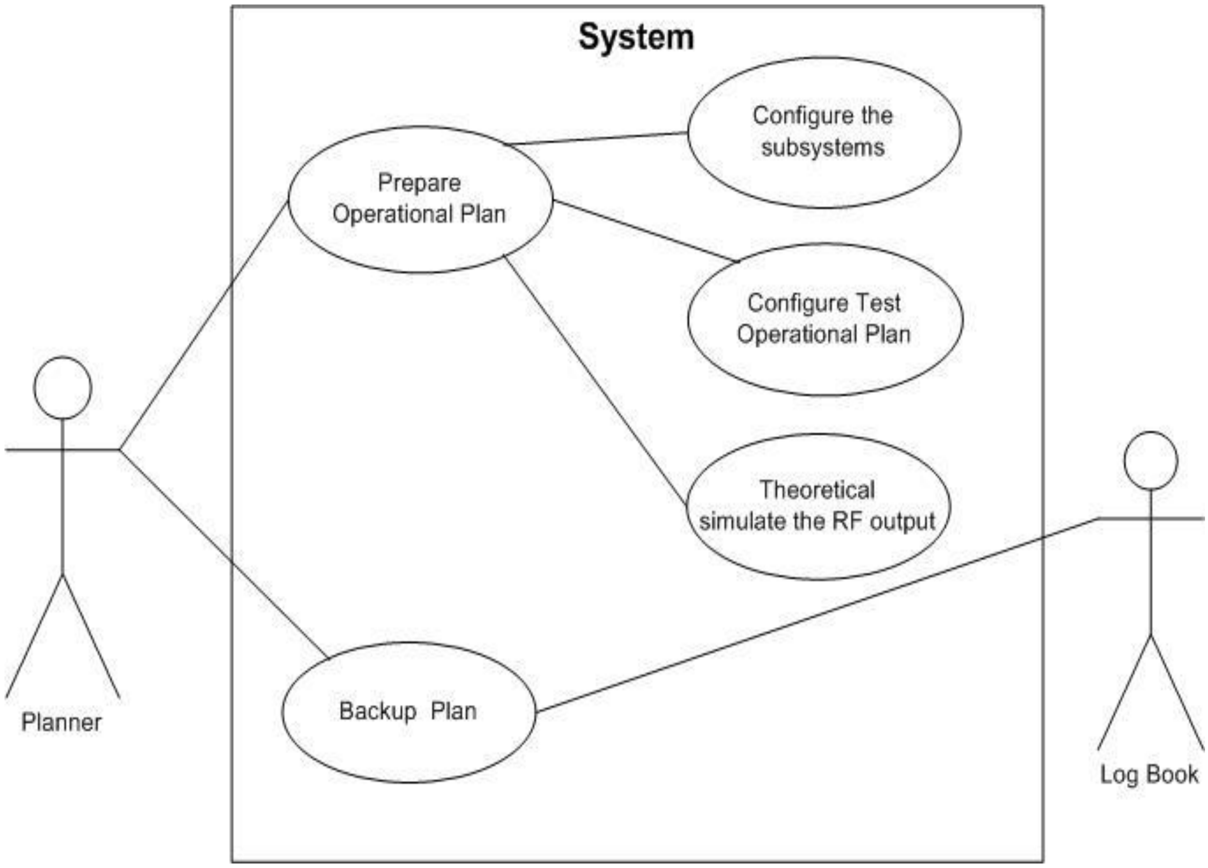


Fig. 2.1: Experiment Planner Module Use Case Diagram

2.4.2 Experiment Execution Requirements

The Experiment operation refers to making various subsystems to boot to the preconfigured values and then supervising the whole process on a console. After the system booting and configuring to planned values, it is required to operate the system as per the test plans one by one. A use case diagram for the experiment execution requirements is shown in Figure 2.2. Operation includes following purposes as below.

1. Show the experiment plan to be executed
2. Set the value of various system components as per the planned experiment
    a. Booting of Magnet Power Supply 1
    b. Booting of Magnet Power Supply 2
    c. Booting of Magnet Power Supply 3
    d. Booting of Filament Power Supply
    e. Booting of Anode Modulator Power Supply
    f. Booting of High Voltage Power Supply

g. Booting of Ion Pump Power Supply

h. Test Case – SST-1 Tokamak

3. Monitor the status of the system components continuously
4. Execution of the experiment through different test cases
5. Monitor the output produced

Post Condition:

The outputs for different test cases are registered.

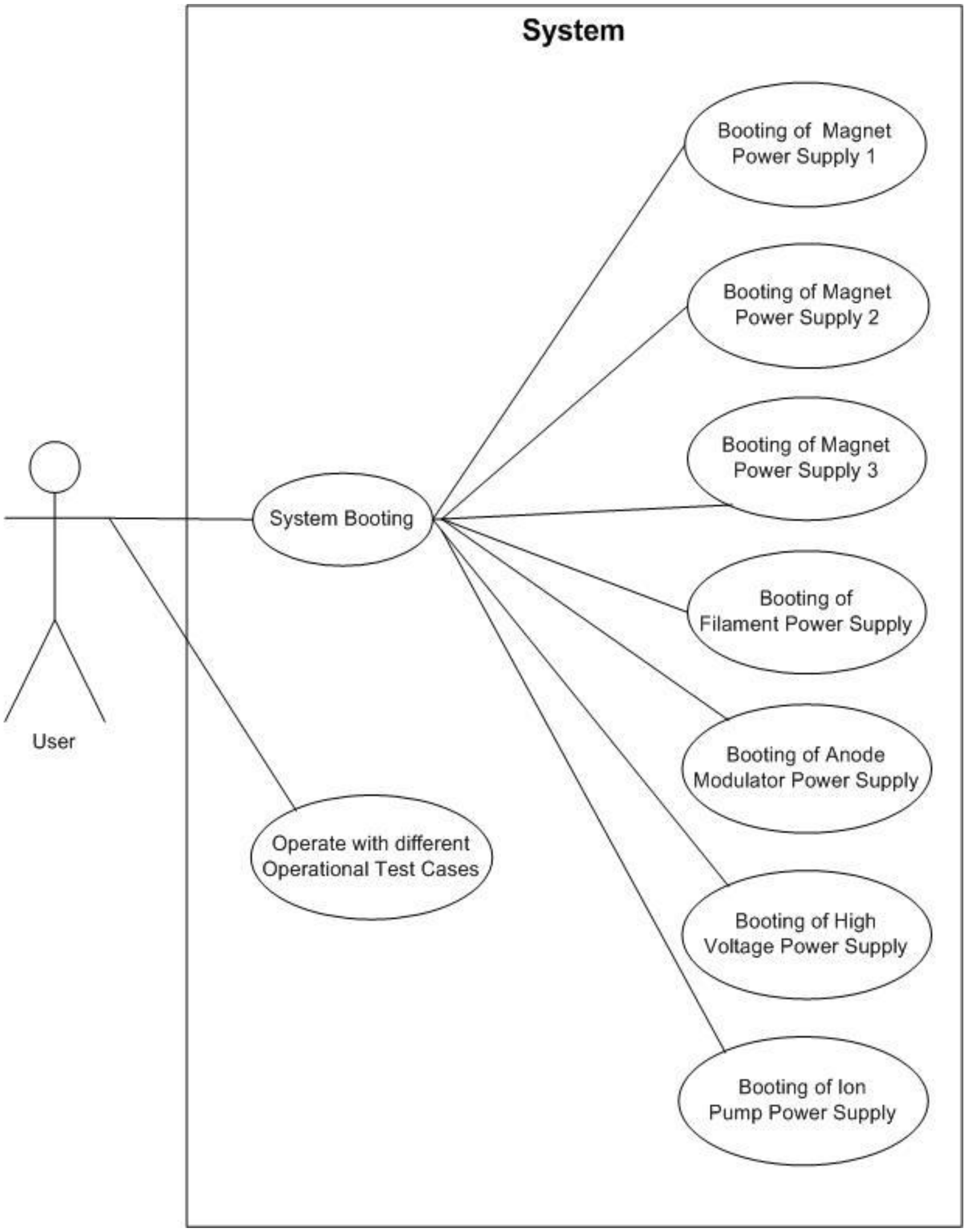


Fig. 2.2: Experiment Execution Module Use Case Diagram

2.4.3 Data Visualization Requirements

Data visualization referred to publish the data onto the web so that the authorized users can get the result and check the performance of various subsystems. A use case diagram for the data visualization requirements is shown in Figure 2.3. Visualization includes following purposes as below.

1. View the experimental log specific to a particular date
2. View the other details associated with a particular experiment like activity information.
3. View the individual graph regarding to a specific Time Dimensional attribute as well as the display of all graphs at a time for the purpose of analysis.

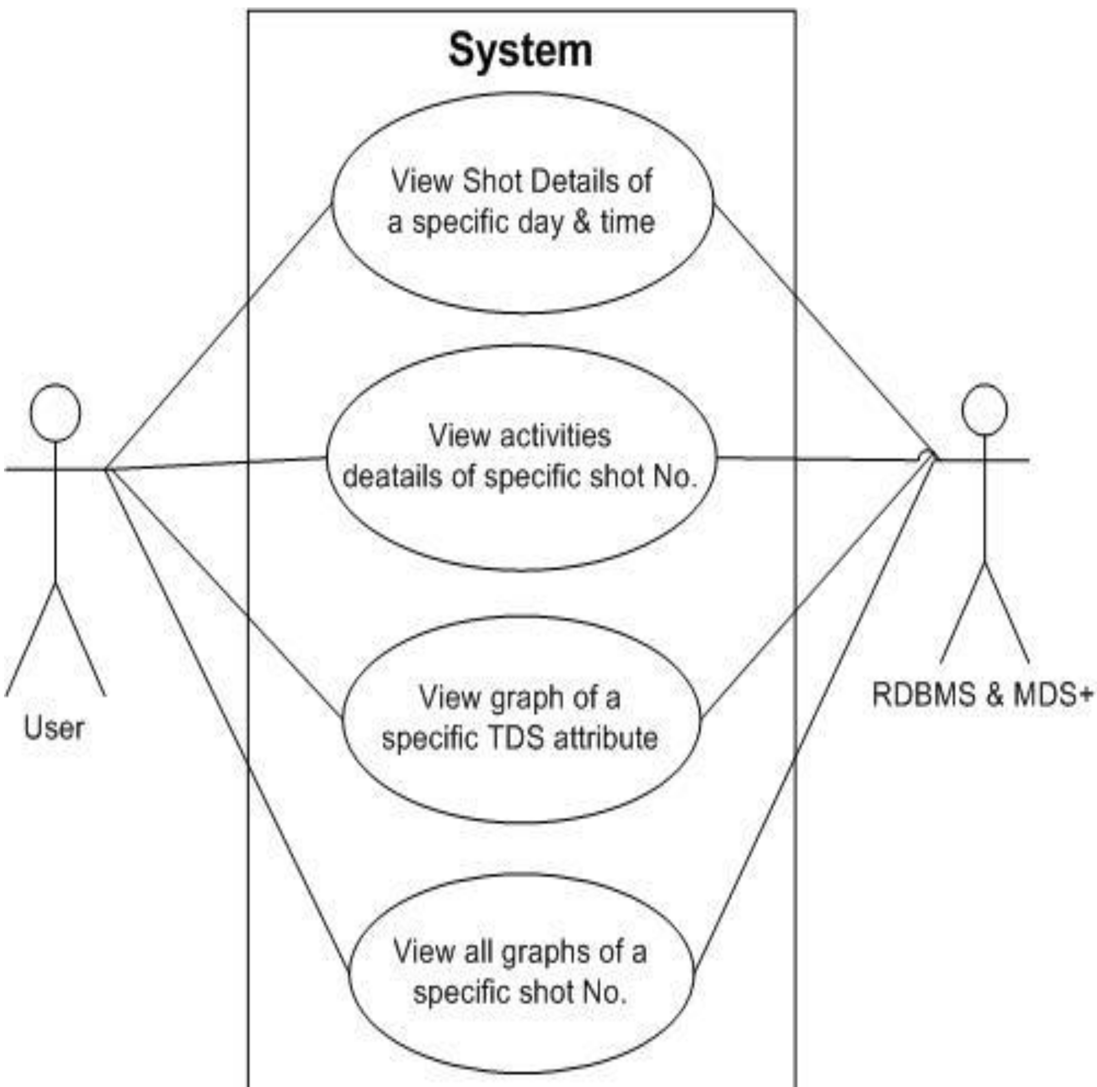


Fig 2.3: Experimental Analyzer Module Use Case Diagram

Post Condition

The system performance groups in different test cases are presented in the graphical format.

The requirements addressed in this chapter are used to design the required system. Before designing the system, sufficient literature study is carried out to find the most suitable architecture and the technology for implementing the project so that the need and the objectives of the system can be fulfilled .The literature study is explained in chapter 3.

3. LITERATURE STUDY

Main Objective of this chapter is to explain the basic concept of Service Oriented Architecture (SOA), its entities and the characteristics. This chapter also addresses Web Services which is used to implement Service Oriented Architecture. It explains the features of Web Services, Web Service Technology Stack and the technologies of Web Services.

3.1 SERVICE ORIENTED ARCHITECTURE

A Service Oriented Architecture is a collection of services. These services communicate with others. The communication can involve either simple data passing or it could involve two or more services coordinating some activities. Service-oriented architectures provide a promising way to address problems related to the integrations of heterogeneous applications in a distributed environment [3] which is mainly required in this project. The World Wide Web consortium [4] defines SOA as below.

A Service Oriented Architecture (SOA) is a form of distributed systems architecture that is typically characterized by the following:

> **Logical View:** The service is an abstracted, logical view of actual programs, databases, business processes, etc., defined in terms of what it does, typically carrying out a business-level operation.
>
> **Message Orientation:** The service is formally defined in terms of the messages exchanged between provider agents and requester agents, and not the properties of the agents themselves. The internal structure of an agent, including features such as its implementation language, process structure and even database structure, are deliberately abstracted away in the SOA: using the SOA discipline one does not and should not need to know how an agent implementing a service is constructed. A key benefit of this concerns so called legacy systems. By avoiding any knowledge of the internal structure of an agent, one can incorporate any software component or application that can be "wrapped" in message handling code that allows it to adhere to the formal service definition.

Description Orientation: A service is described by machine-processable meta data. The description supports the public nature of the SOA: only those details that are exposed to the public and important for the use of the service should be included in the description. The semantics of a service should be documented, either directly or indirectly, by its description.

Granularity: Services tend to use a small number of operations with relatively large and complex messages.

Network orientation: Services tend to be oriented toward use over a network, though this is not an absolute requirement.

Platform neutral: Messages are sent in a platform neutral, standardized format delivered through the interfaces. XML is the most obvious format that meets this constraint.

The mentioned definition is very generic and the following definition can be adopted: In SOA, software applications are built on basic components called services. A service in SOA is an exposed piece of functionality with three properties:

1. The interface contract to the service is platform independent.
2. The service can be dynamically located and invoked.
3. The service is self-contained. That is, the service maintains its own state.

So SOA is a special kind of a software architecture that has several unique characteristics. Most important aspect of SOA is that it separates the service's implementation from its interface. In other words, it separates the "what" from the "how". Service consumer view a service simply as an endpoint that supports a particular request format or contract. Service consumers are not concerned with how the service goes about executing their requests; they expect only that it will. Consumer also expect that their interaction with the service will follow a contract, in an agreed upon interaction between two parties.

3.1.1 SOA Entities

SOA consists of the following 6 entities [5]. The various entities are shown in the Figure 3.1 and they are explained as follows:

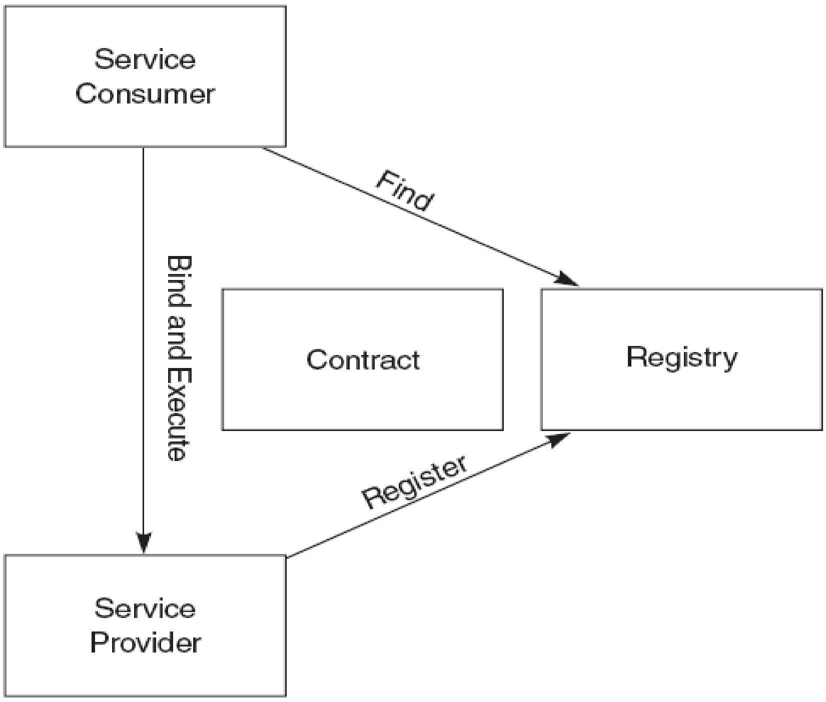


Fig 3.1: The "find-bind execute" Paradigm in SOA.

1. ***Service Consumer:*** It is an application, service or some other type of software module that requires a service. It is the entity that initiates the locating of the service in the registry, binding to the service over a transport, and executing the service function. The service consumer executes the service by sending it a request formatted according to a contract.

2. ***Service Provider:*** It is the service; the network addressable entity that accepts and execute requests from consumers. It can be a mainframe system, a component or some other type of software system that executes the service request. The service provider publishes its contract in the registry for access by service consumers.

3. ***Service Registry:*** It is a network based directory that contains available services. It is an entity that accepts and stores contracts from service providers and provides that contract to interested service consumers.

4. ***Service Contract:*** It is the specification of the way a consumer of a service will interact with the provider of the service. It specifies the format

of the request and response from the service. A service contract may require a set of preconditions and post conditions. These preconditions and post conditions specify the state that the service must be in to execute a particular function. The contract may also specify the Quality of Service (QoS), which are the nonfunctional aspects of the service.

5. ***Service Proxy:*** The service provider supplies a proxy to service consumer as shown in Figure 3.2. The service consumer executes the request by calling an API function on the proxy. The service proxy finds a contract and a reference to the service provider in the registry. It then formats the request message and executes the request on behalf of the consumer. It is optional that enhances the performance by catching remote references and data. When a proxy catches a remote reference, subsequent service calls will not require additional registry calls. By storing service contracts locally, the consumer reduces the network hops required to execute the service.

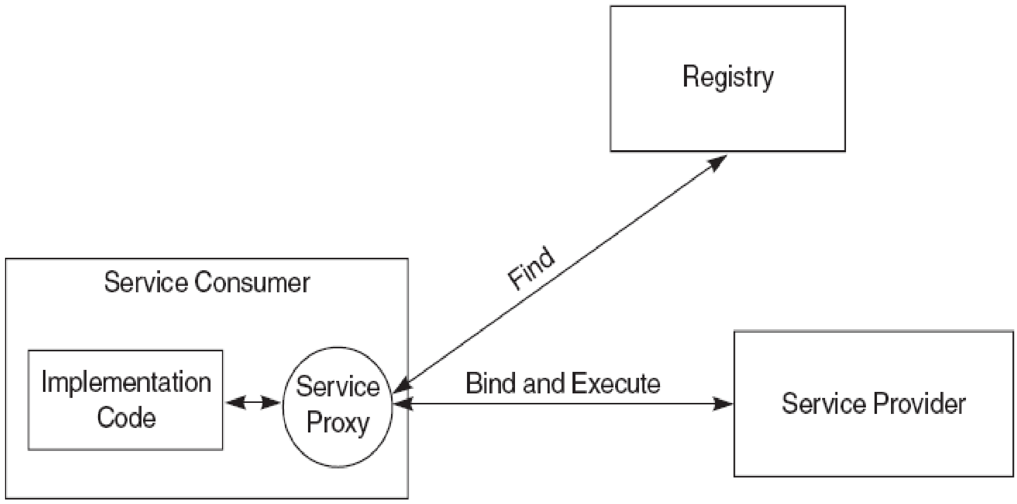


Fig 3.2: Service Proxy Entity

6. ***Service Lease:*** It specifies the amount of time the contract is valid, only from the time the consumer request it from the registry to the time specified by the lease. When the lease runs out, the consumer must request a new lease from the registry. It is necessary to maintain state information about the binding between the consumer and the provider.

3.1.2 SOA Characteristics

SOA characteristics [5] can be summarized as follows:

Services are discoverable and dynamically bound.
Services are self contained and modular.
Services stress interoperability.
Services are loosely coupled.
Services have a network addressable interface.
Services have coarse grained interfaces.
Services are location transparent.
Services are composable.
SOA supports self healing.

3.1.3 Quality Of Services In SOA

Quality attributes like interoperability, reliability, performance, security etc drive the design of architecture. This section describes these implications for a set of quality attributes in the context of an SOA. These are also called service's nonfunctional aspects. [6]

Interoperability: Interoperability refers to the ability of a collection of communicating entities to share specific information and operate on it according to an agreed-upon operational semantics. Through the use of the underlying standards, an SOA provides good interoperability technology-wise overall.

Reliability: It is the quality aspect of the web service that represents the degree of being capable of maintaining the service and service quality. The number of failures per month or year represents a measure of reliability of a web service. Several aspects of reliability are important within an SOA, particularly the reliability of the messages that are exchanged between the application and the services, and the reliability of the services themselves. Applications developed by different organizations may have different reliability requirements for the same set of services. An application that operates in different environments may have different reliability requirements in each one. Potentially, problems can occur in many areas, but the use of the underlying standards (WS-Reliability and WS-Reliable Messaging) should mean that messages are transmitted reliably.

Security : Although security denotes different things with respect to software systems, in general, it is associated with four principles:

Confidentiality – Access to information/service is granted only to authorized subjects.

Authenticity – We can trust that the indicated author/sender is the one responsible for the information.

Integrity – Information is not corrupted.

Availability – The information/service is available in a timely manner.

Security is a major concern for SOA and Web services. The need for encryption, authentication, and trust within an SOA approach requires detailed attention within the architecture. Many standards are being developed to support security, but most are still immature.

Performance: Performance is the quality aspect of Web Service, used for implementing SOA, which is measured in terms of throughput and latency. Web Services are described in next section 3.2. Higher throughput and lower latency values represent good performance of a Web Service. Throughput represent the number of Web Services requests served at a given time period. Latency is the round trip time between sending a request and receiving the response. An SOA approach can have a negative impact on the performance of an application due to network delays, the overhead of looking up services in a directory, and the overhead caused by intermediaries that handle communication.

3.2 WEB SERVICES

Web services allow anybody to create electronic services that can be used by anyone else, so enabling electronic anybody to anybody communications, coordination, and interaction. Web services are nothing but two programs interchanging data on the Internet or an intranet.

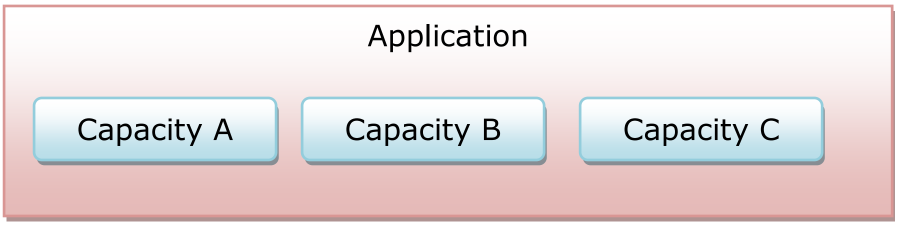
Web Services are pieces of application functionally or business processes that are available for use by other applications- either internally to the enterprise or externally with other companies and organizations. Any application can use a web service by simply addressing it using its URL and invoking it across a network. [6]

The W3C defines a Web service as a software system designed to support interoperable machine-to-machine interaction over a network.

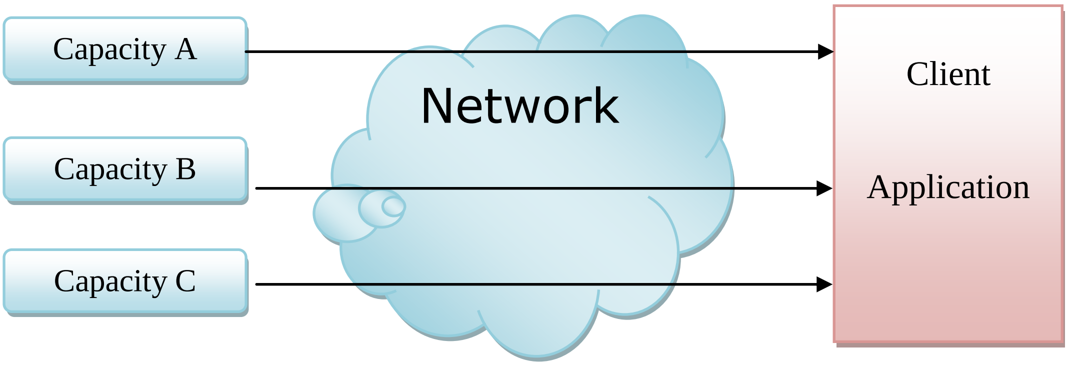
3.2.1 Salient features of Web Services

Web Services expose their capabilities to client applications. Not their implementations. This allows Web services to be implemented in any language and on any platform and still be compatible with all client applications.

Each Web Service is self contained: it describes its own capabilities, publishes its own programmatic interface, and implements its own functionality that is available as a hosted service.



Monolithic application with integrated capabilities A, B and C



Client Application invoking remote Web Services for Capabilities A,B, & C

Fig 3.3: Example of a Monolithic Application and a Web Application

Web Services offer capabilities that can by used by client applications by invoking it across a network using its URL. The client application does not have to integrate the Web Services, but simply uses their capabilities as remote services.

The business logic of a Web Service runs on a remote machine that is accessible by other applications through a network. The client application simply invokes the functionality of a Web Service by sending it messages, receives return messages from the Web Service, and then uses the results within the application. There is no need to integrate the Web Service within the client application into as single monolithic block. This reduces development and testing times, maintenance cost, and overall errors. [6]

3.2.2 Web Service Technology Stack

Web Service Technology Stack is a collection of standardized protocols and application programming interfaces (APIs) that lets individuals and applications locate and utilize Web services. After introducing the stack itself, it is illustrated how each of its layers facilitates the use of Web services in section 3.2.3.

<table>
<tr><th>Layer Description</th><th>Implementation(s)</th><th colspan="4">Other Concerns</th></tr>
<tr><td>Standard Messaging</td><td>Electronic Buisness XML initiative (ebXML)</td><td rowspan="6">Quality of Service</td><td rowspan="6">Management</td><td rowspan="6">Security</td><td rowspan="6">Service Development</td></tr>
<tr><td>Service composition</td><td>Buisness Process Execution Service for Web Services (BPEL4WS)</td></tr>
<tr><td>Service Registry</td><td>Universal Description, Discovery and Integration (UDDI) ebXML Registers</td></tr>
<tr><td>Service Description</td><td>Web Service Description Language (WSDL)</td></tr>
<tr><td>Service Messaging</td><td>Simple Object Access Protocol (SOAP) / Extensible Markup Language (XML)</td></tr>
<tr><td>Service Transport</td><td>Hyprertext Transfer Protocol (HTTP)
Simple Mail Transfer Protocol (SMTP)
File Transfer Protocol (FTP)</td></tr>
</table>

Fig 3.4: Web Service Technology Stack

3.2.3 The Technologies of Web Services

The technologies that form the foundations of Web Services are SOAP, WSDL, and UDDI. [7] The relationship between SOAP, WSDL, and UDDI in publishing, discovering and invoking Web Services is shown in figure 3.5.

SOAP: Simple Object Access Protocol (SOAP) is an XML-based mechanism for exchanging information between applications within a distributed environment. This information exchange mechanism can be used to send messages between applications, and more specifically, can be used to implement remote procedure calls (RPCs). RPCs allow one application to invoke and use a procedure of another application.

WSDL: Web Services Description Language (WSDL) is an XML-based language for describing Web Services. Through a WSDL description, a client application can determine the location of the remote Web Services, the function it implements, as well as how to access and use each function. After parsing a WSDL description, a client application can appropriately format a SOAP request and dispatch it to the location of the Web Service.

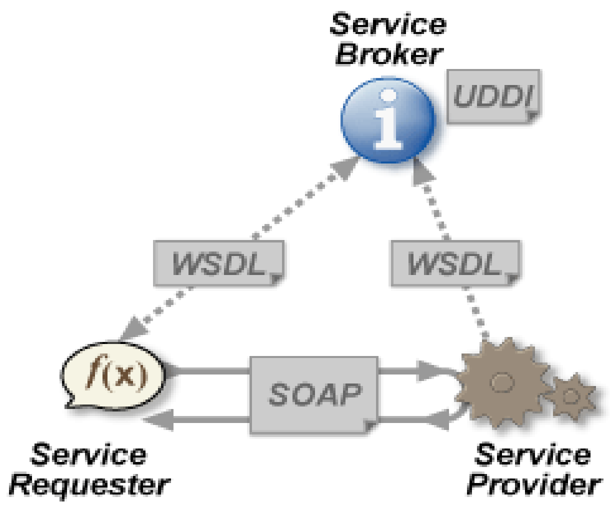


Fig 3.5 Relationship between SOAP, WSDL, and UDDI in Web Services

UDDI: Universal Description, Discovery and Integration (UDDI) is a specification for a registry of information for Web Services. UDDI defines a means to publish and more importantly, discover (or search for) information, including WSDL files about Web Services.

After browsing through an UDDI registry for information about available Web Services, the WSDL for the selected service can be parsed, and an appropriate SOAP message can be sent to the service.

3.2.4 Importance o Web Services:

Web Services are indeed a technology for distributed computing. One critical difference between these two is that a person who implements a Web Service can be almost one hundred percent certain that anybody else can communicate with and use the service.
Based on industry standards such as XML and HTTP, and supporting near ubiquitous interoperability, Web Services places few restrictions on with which other applications can interact.

In nutshell, this chapter finds Service Oriented Architecture and Web Services as the most appropriate architecture and technology to implement the required project. In addition, Netbeans 5.5 Integrated Development Environment (IDE) is used to develop the required modules and Sun Java Application Server Platform Edition 9.0 is used for deploying and hosting the web services. The salient features and the reason to use these tools are explained in detail in Appendix-A and Appendix-B respectively.

4. SYSTEM ANALYSIS AND DESIGN

This chapter presents system analysis and design of the proposed system in detail. First, proposed system architecture is described here. In addition to this, the designs of the required modules are explained with the help of sequence diagram and collaboration diagram. All the requirements specified in chapter 2 are addressed here.

4.1 PROPOSED SYSTEM

This project has been planned and designed to be modular in nature. The integration layer is proposed by this project so that plug and play based application integration approach can be adapted. The LHCD DAC system is of three tier in nature. In first tier, the hardware and I/O devices are handled under real time operating system based applications. The focus point of the project provides the middle and front tier. In the middle tier, SOA based application layer compromised of Application server and J2EE tools has been designed which provides the secure and authenticate way to access the first tier. The purpose of front tier is to provide the faces for user interaction. It is developed in Java Server Pages (JSP). The remote user is concerned only for the services it required not for the implementation. The beauty of the project lies in distributive nature of all the three tiers. The proposed system architecture is shown in Figure 4.1 which consists of all the three layers mentioned above.

The project plan was to provide following modules at middle and front tier.

Steady State High Power RF system at 3.7 GHz Experiment Planner Module

Steady State High Power RF system at 3.7 GHz Experiment Executor Module

Steady State High Power RF system at 3.7 GHz Experiment Analyzer Module

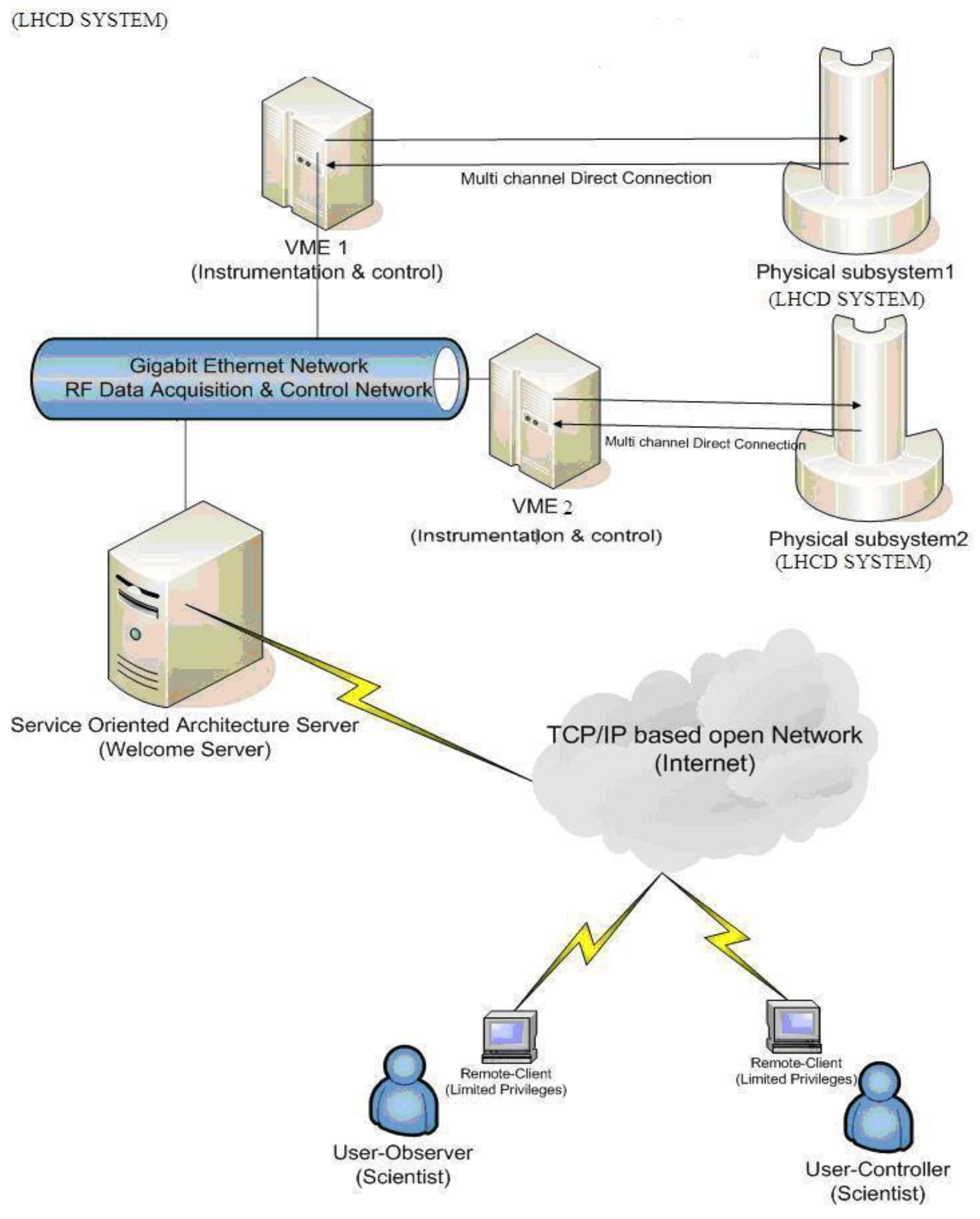


Fig 4.1: Proposed System Architecture

4.2 EXPERIMENT PLANNER MODULE DESIGN

The design of experimental planner module is described by sequence and collaboration diagram. Sequence diagrams are mainly used to describe the signaling and message passing between objects. Collaboration diagram, also known as interaction diagram, illustrates the relationship and interaction between software objects. The sequence and interaction of the operation for the experiment planner module is shown in diagram 4.2 and 4.3 respectively.

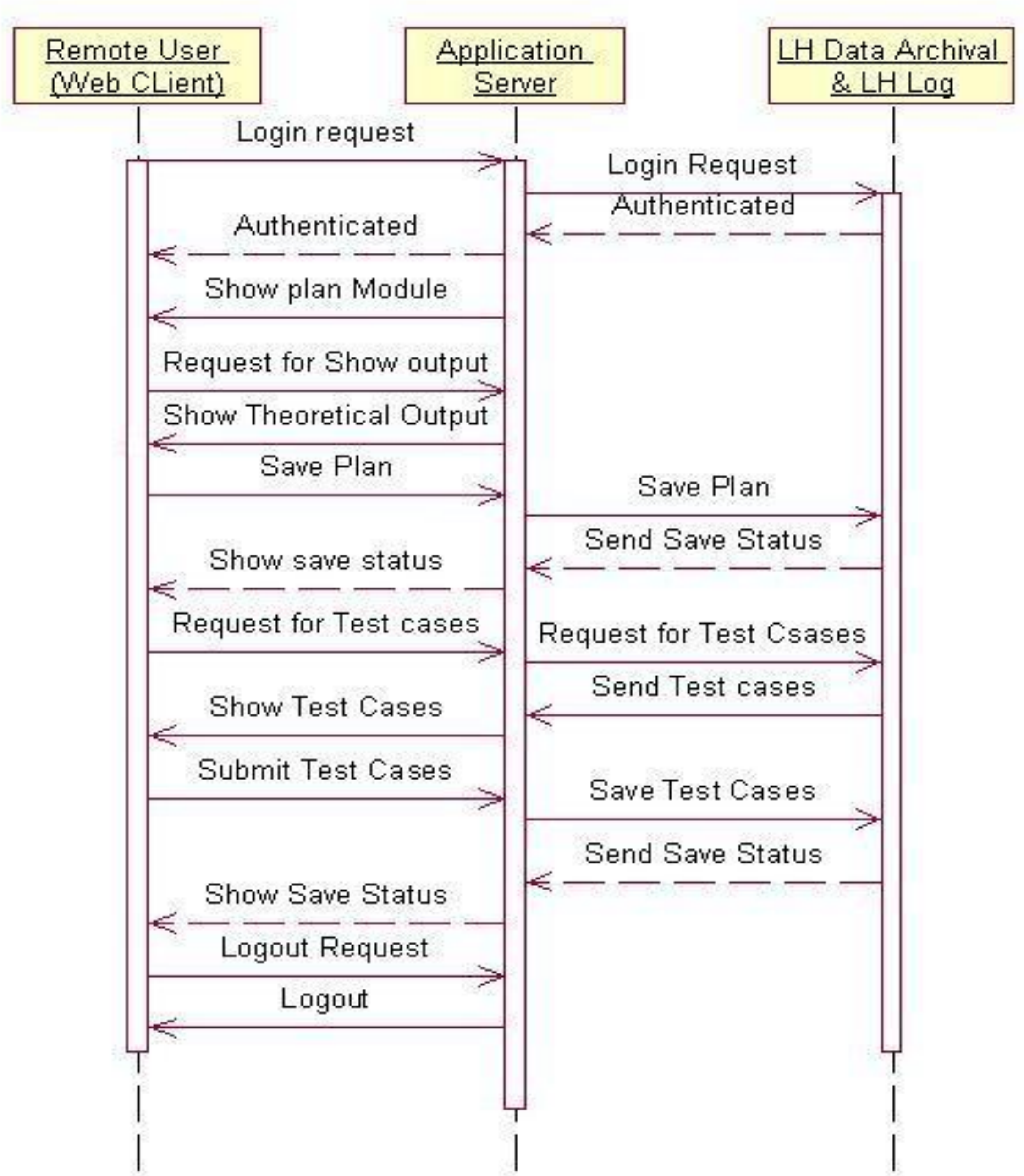


Fig 4.2: Experimental Planner Module Sequence Diagram

Description:

The sequence diagram shown as in Figure 4.2 describes how experiment planner module works. After successful login, remote user views the planner module page. Then the purpose and certain input parameters according to the purpose selected are supplied by the remote user. After submitting the plan, a theoretically simulated output is shown to the planner which is based on the literature and previous experiment results. After being satisfied with the output shown, planner is asked for saving the plan. When a plan is saved, planner is asked to associate test cases with this plan and a test case window is shown to the planner. A planner can submit a number of test cases with a particular plan. He can add, delete and modify the test cases at the time of entry. Later on, the test cases are saved. Finally, the planner exits from the planner module.

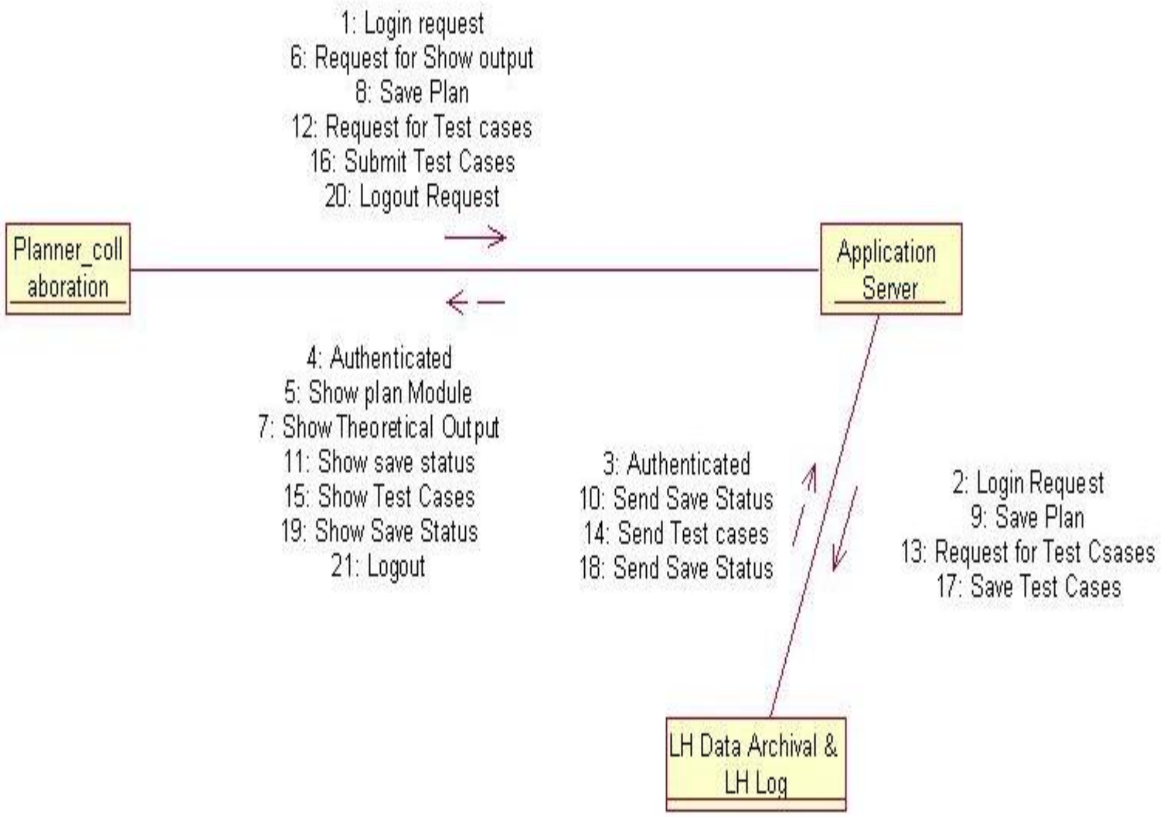


Fig 4.3: Experimental Planner Module Collaboration Diagram

4.3 EXPERIMENT EXECUTOR MODULE DESIGN

The sequence of the operations and interaction with different entities for the experiment executor module are shown in Figure 4.4 and 4.5 respectively.

Description:

As shown in the sequence diagram below, after successful login, remote executor views the executor module page. First, objective of the experiment is requested. The previously planned experiment with associated test cases is shown. The facility for test cases navigation is also provided. The status of every component (ON, READY, TRIP) and the value of characteristic signal (voltage and current) are refreshed within each 500 milliseconds. It is done without any remote executor intervention. The values of various components can be set through the web page. A simulated theoretical output for particular input is shown immediately. System status is regularly updated automatically and it is observed and closely monitored. Whenever trip occurs, the functioning of component is stopped. It can be stopped manually by remote executor at any time. The starting sequence of booting of components is done in a particular order while stopping can be done in any order. Now, after setting values, test cases are operated and the output is shown in the respective area.

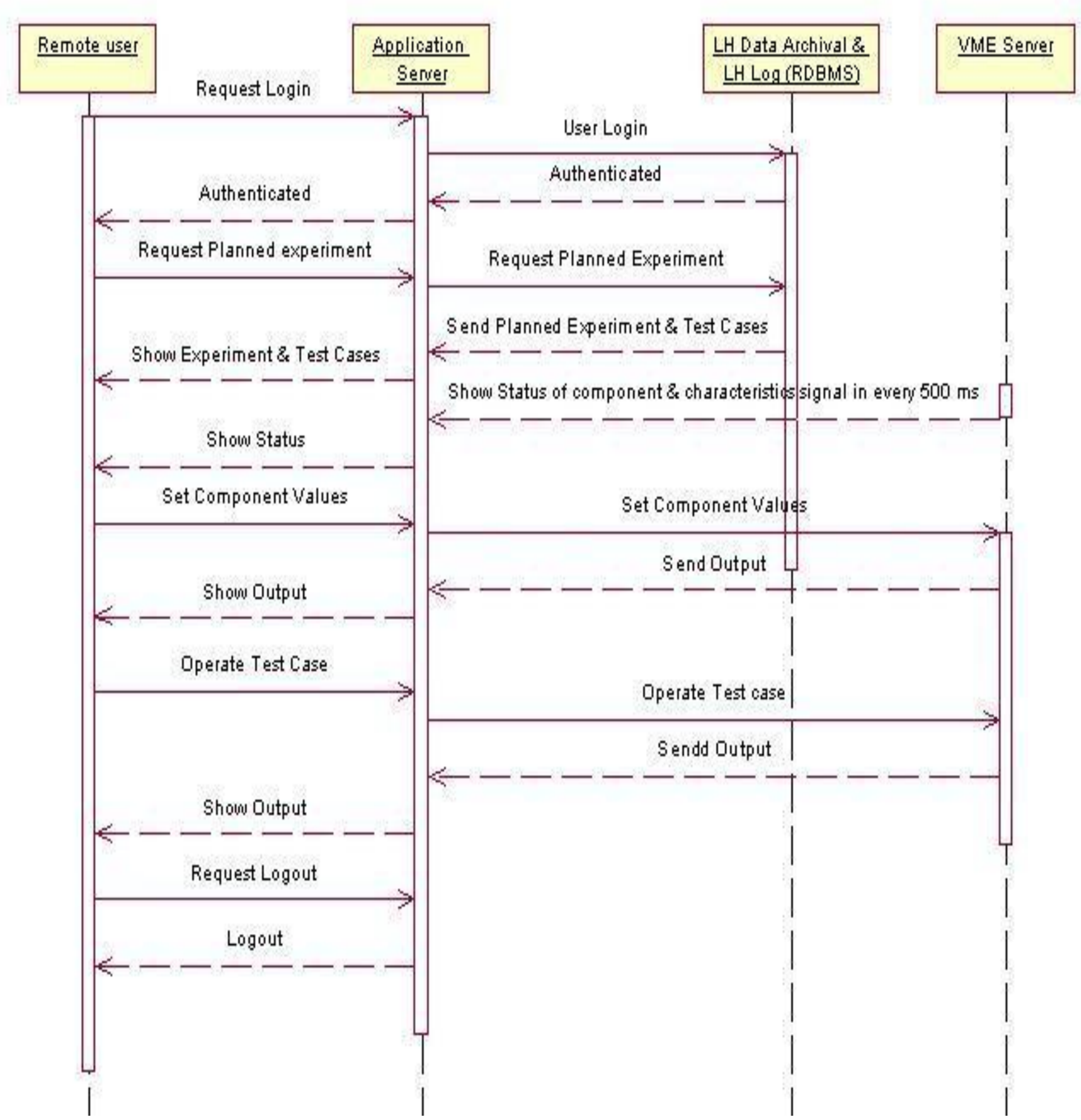


Fig 4.4: Experimental Executor Module Sequence Diagram

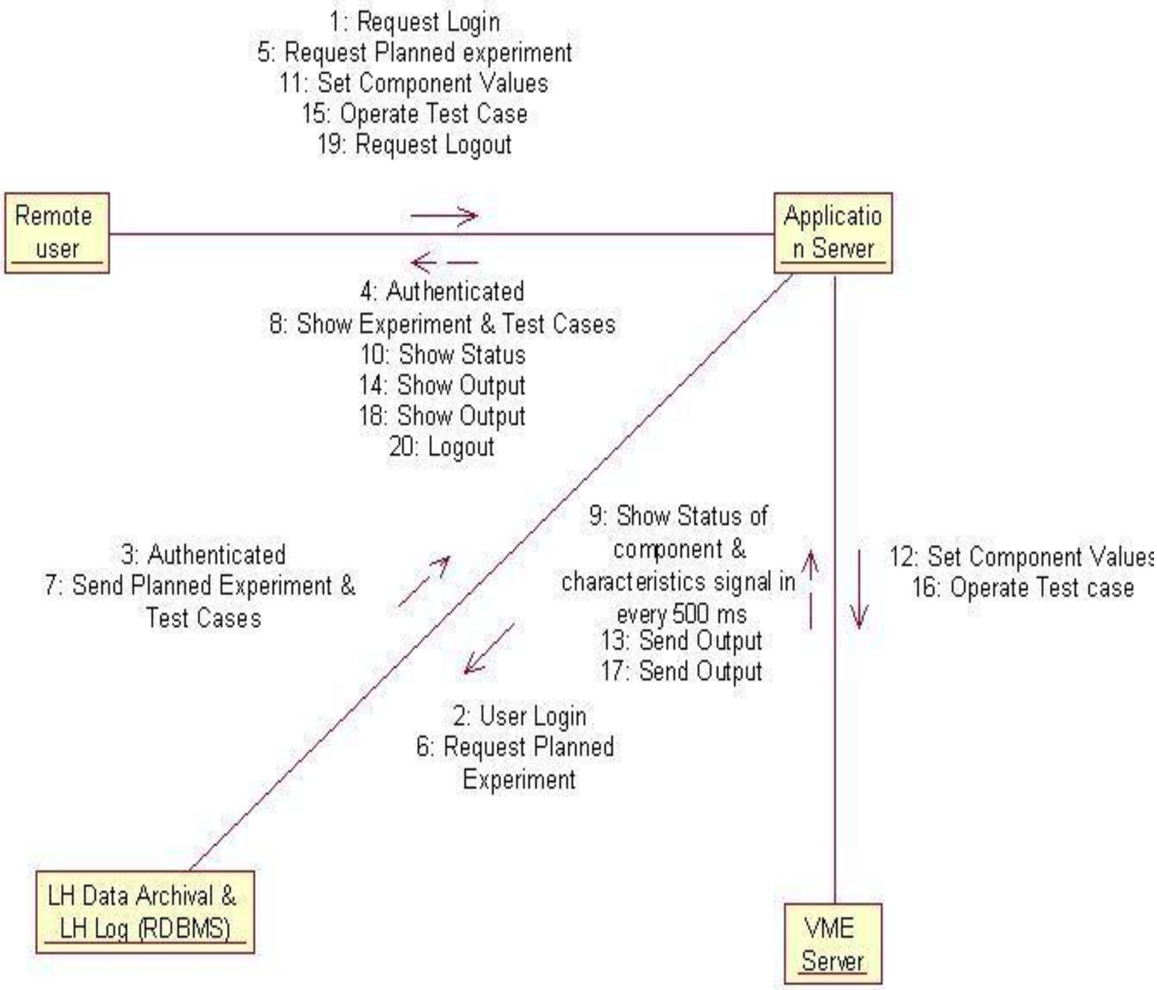


Fig 4.5: Experimental Executor Module Collaboration Diagram

EXPERIMENT ANALYZER MODULE DESIGN

During analysis phase, it is found that there is no current Web based system for data visualization available. There is a graphical tool named **jScope** with which data visualization is possible, but because of its complex management it is used locally; for remote usage there must be a jScope available on the remote machine.

Due to non Web based nature of jScope; customization of graphical representations of data with in one screen is not achievable. The issue of proprietary nature of jScope has also to be taken into consideration when extending the tool to achieve more usability.

So it is required to design a web based Data Visualization System for RF systems that can help the researchers from all around the world to visualize the scientific experimental data in graphical format on the Web. The main objective is to visualize the scientific experiments' data that is stored in tree based database in the form of numeric signal; into graphical format on the Web so that anybody on the intranet or on Internet can view the graphical data and able to see other data associated with signal.

The main design challenge is to enable one to visualize some sort of data on a particular date & particular time for a particular experiment. This objective is only achieved by combining both tree based database and RDBMS. Here, in this project MySQL is used as RDBMS and MDSPlus for tree based database. MDSPlus is described in detail in Appendix-C.

When the user interacts with the system, the system provides the facility to select the values based on the requirements to see the data related with the experiments. By this user can not only view experiment's data but also able to view the other detailed informative data which are related with that experiment. This information is in static form which can be viewed directly. Some information consists of reading taken on particular time, regarding the experiments; for component status. These values are displayed as graph which provides better visualization for analytical purpose.

User can view both individual and grouped graphical representation of experiment components. So at a time more than one graph can be displayed. By this way comparison of the results of two experiments can be done and it can be verified what the problem is there if any, due to experiments.

This is the fundamental need of the system that any one can see the graphs related to the data selected and based on that take decisions on further experiments. The system is designed in the same manner to fulfill all requirements.

The sequence of the operations and interaction with entities for the experiment analyzer module are shown in Figure 4.6 and 4.7 respectively as below.

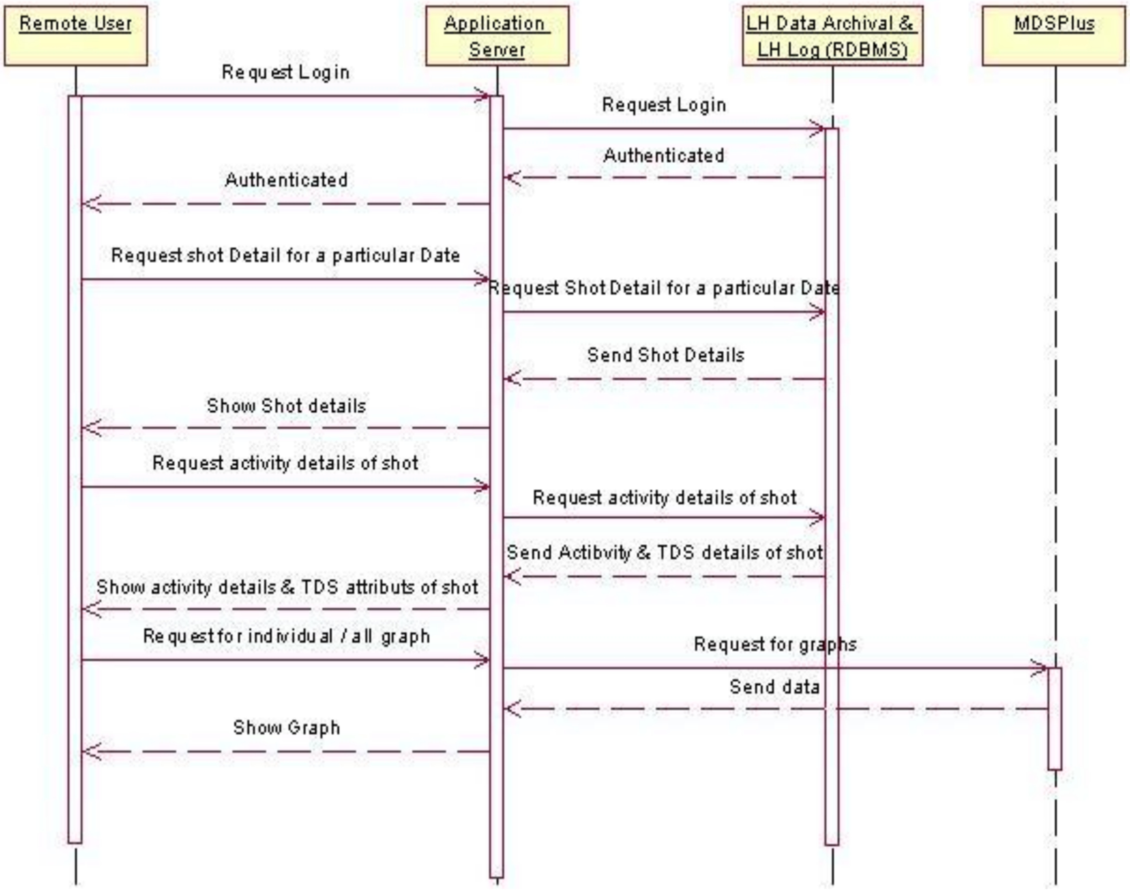


Fig 4.6: Experimental Analyzer Module Sequence Diagram

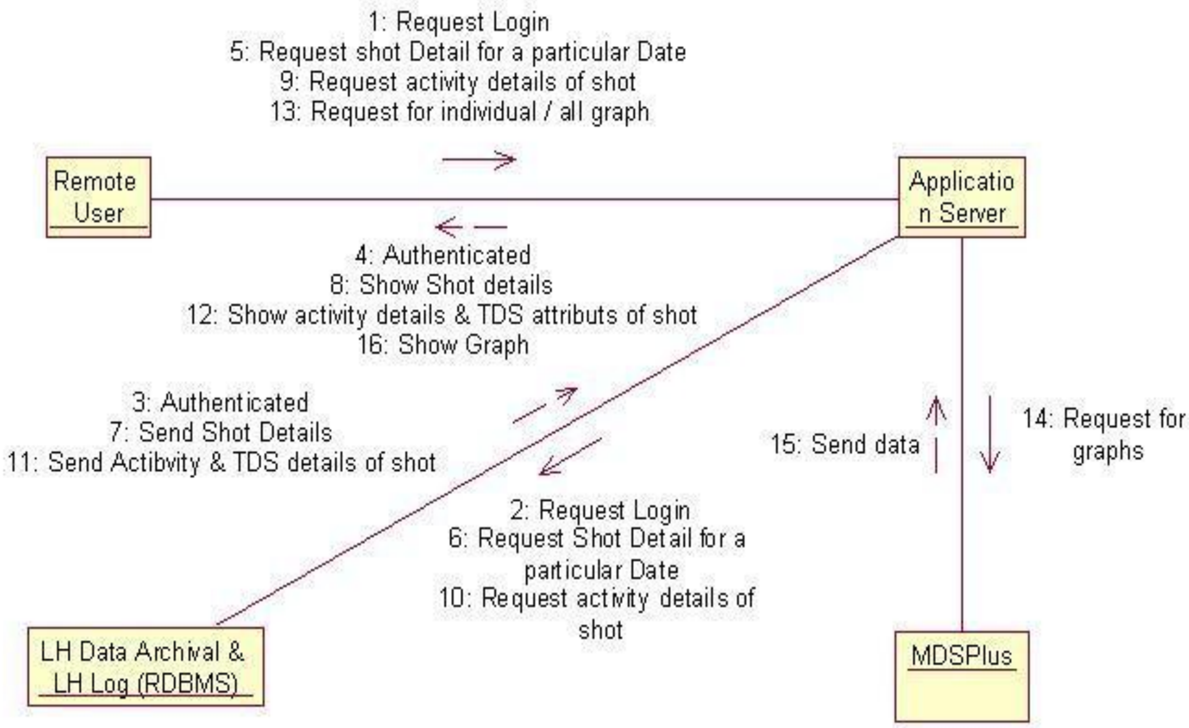


Fig 4.7: Experimental Analyzer Module Collaboration Diagram

4.5 DATABASE DESIGN

There is a need to use relational database to store user information as well as information regarding experiment analyzer module. The various logs related to LH Data like plasma shot details, activities information are maintained through it so that it can provide information against query fired by the remote analyst. The database design diagram is shown as below in Figure 4.8.

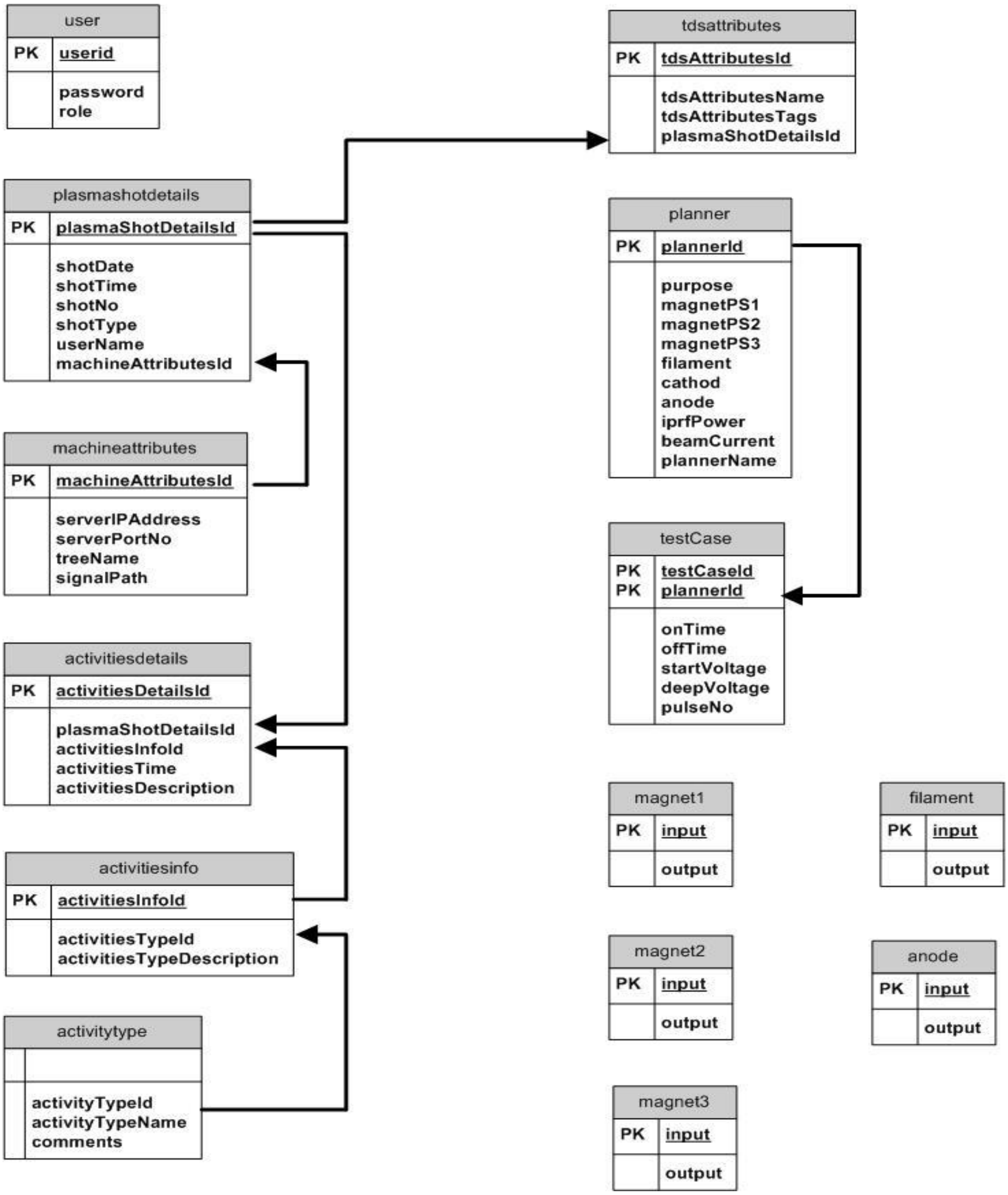


Fig 4.8: Database Design

In this chapter, various design issues are considered for implementation purpose and a final layout for each module is developed. The sequence of the operations and the action to be performed by the various involved entities are finalized. The need and the use of various tools are also clarified in this phase. So this chapter provides a clear direction towards the implementation of the project modules. The output snapshots of the implementations are provided in chapter 5.

5. OUTPUT SNAPSHOTS

In this chapter, output snapshots of the developed modules are shown. It helps to understand how a remote user interacts with the system. User interaction and sequence of operations in each screenshot is explained in Appendix-D in detail.

5.1 ROLE BASED AUTHENTICATION

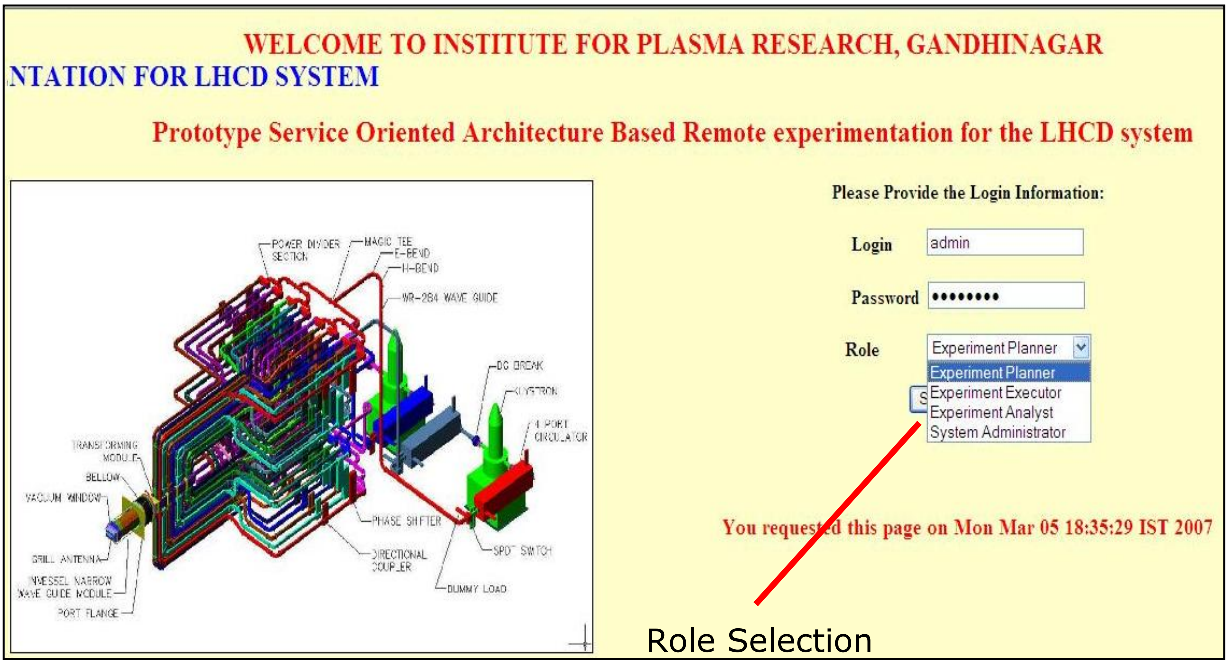


Fig 5.1: Role Based Authentication Screenshot

5.2 EXPERIMENTAL PLANNER MODULE

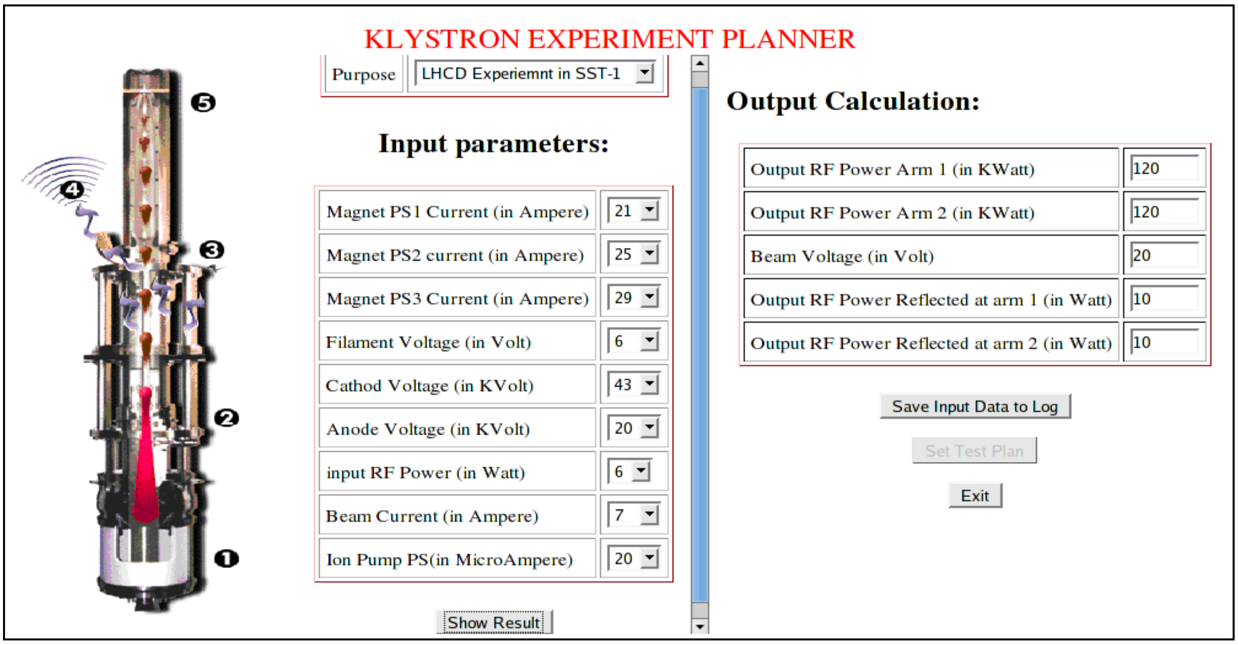


Fig 5.2: Experimental Planner Module Screenshot -1

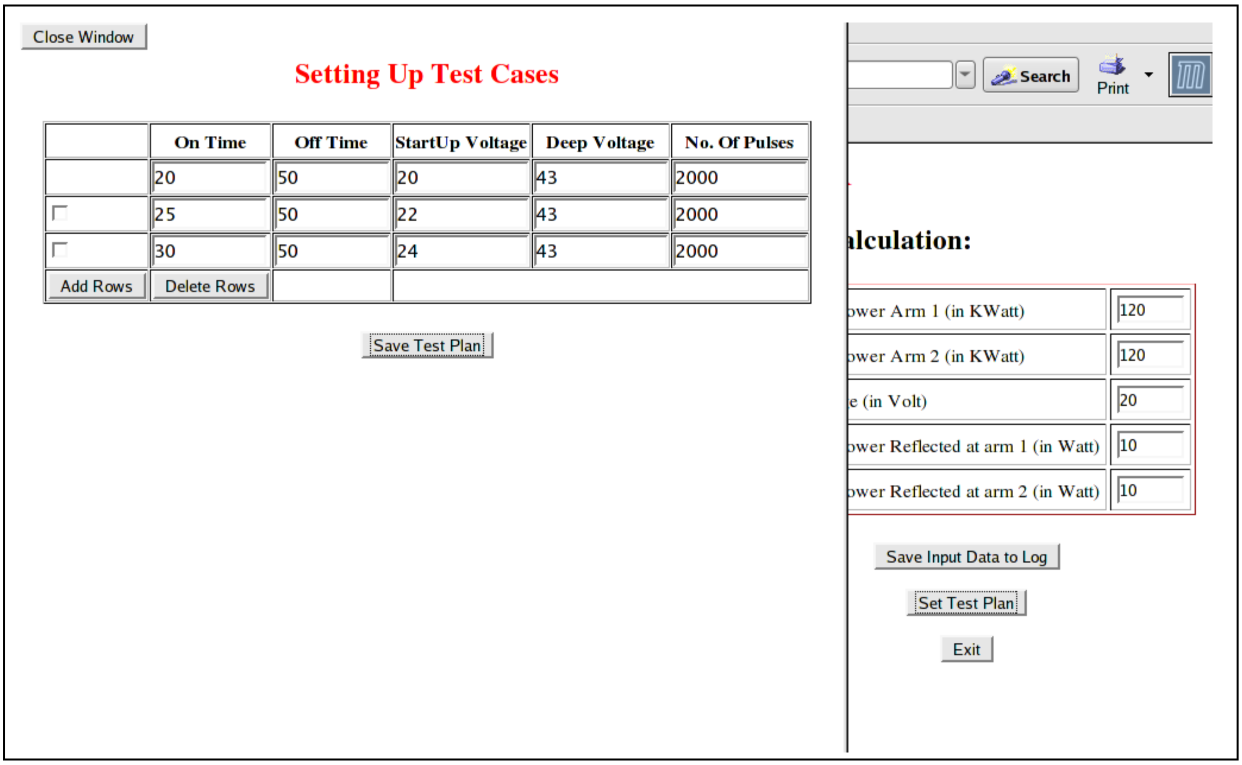


Fig 5.3: Experimental Planner Module Screenshot -2

5.3 EXPERIMENTAL EXECUTOR MODULE

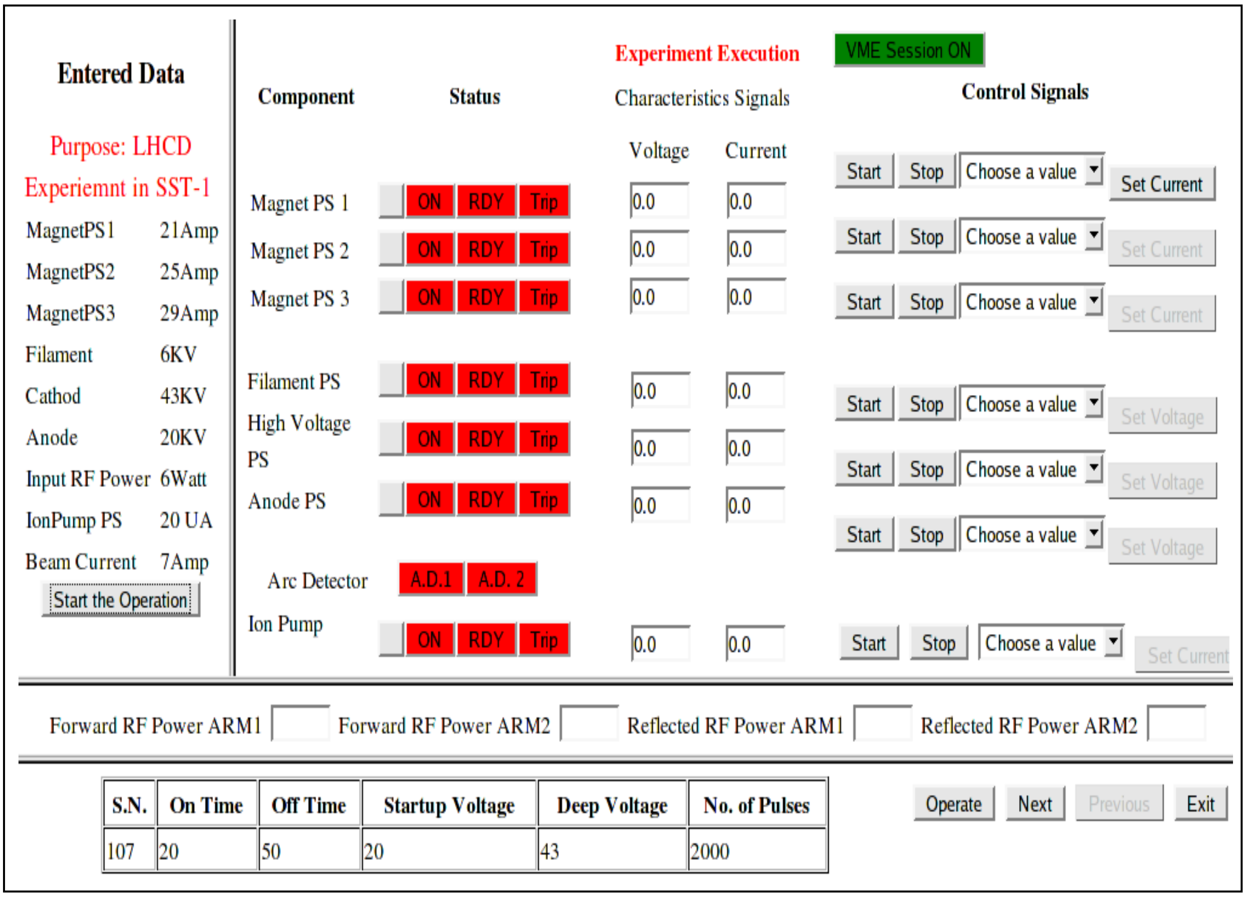


Fig. 5.4: Experiment Executor Module Screenshot -1

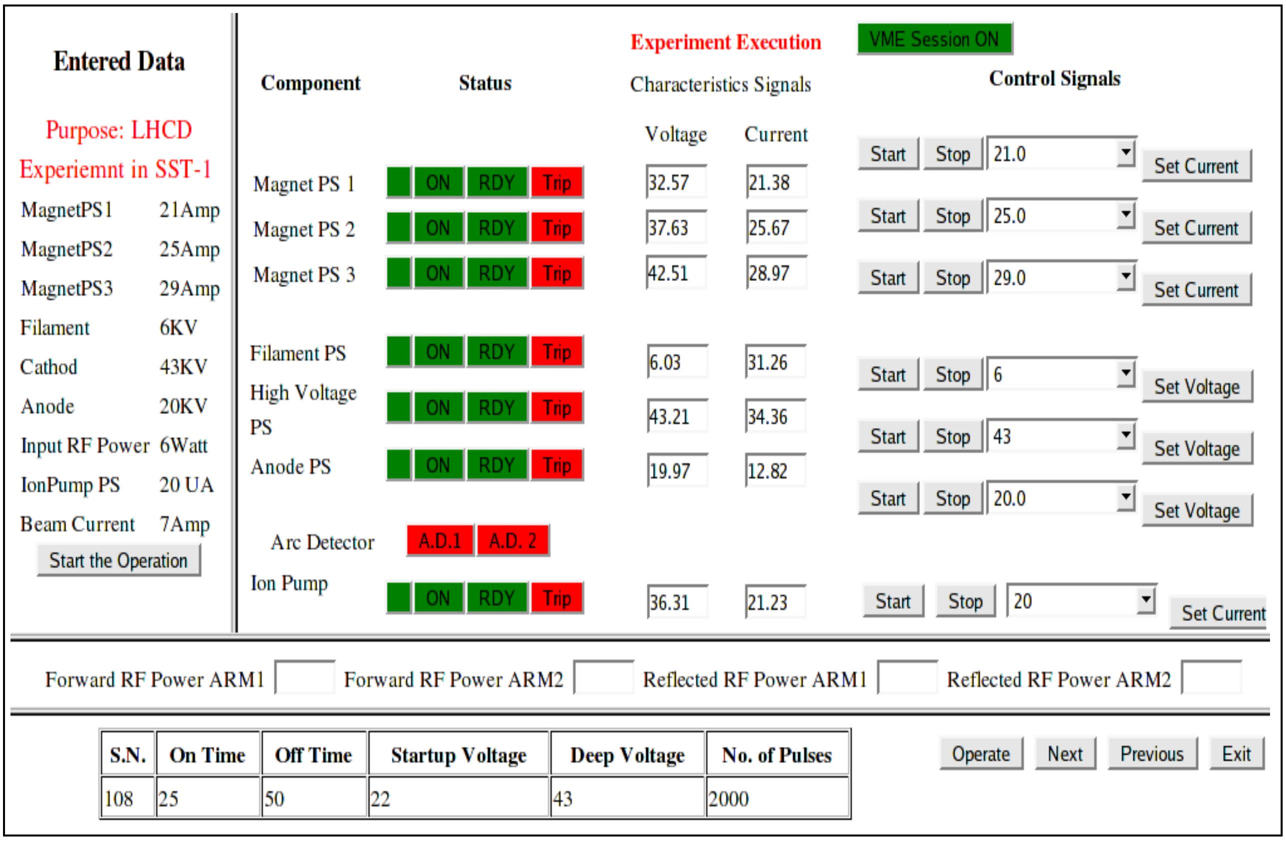


Fig. 5.5: Experiment Executor Module Screenshot -2

5.4 SCREEN SHOT: EXPERIMENTAL ANALYZER MODULE

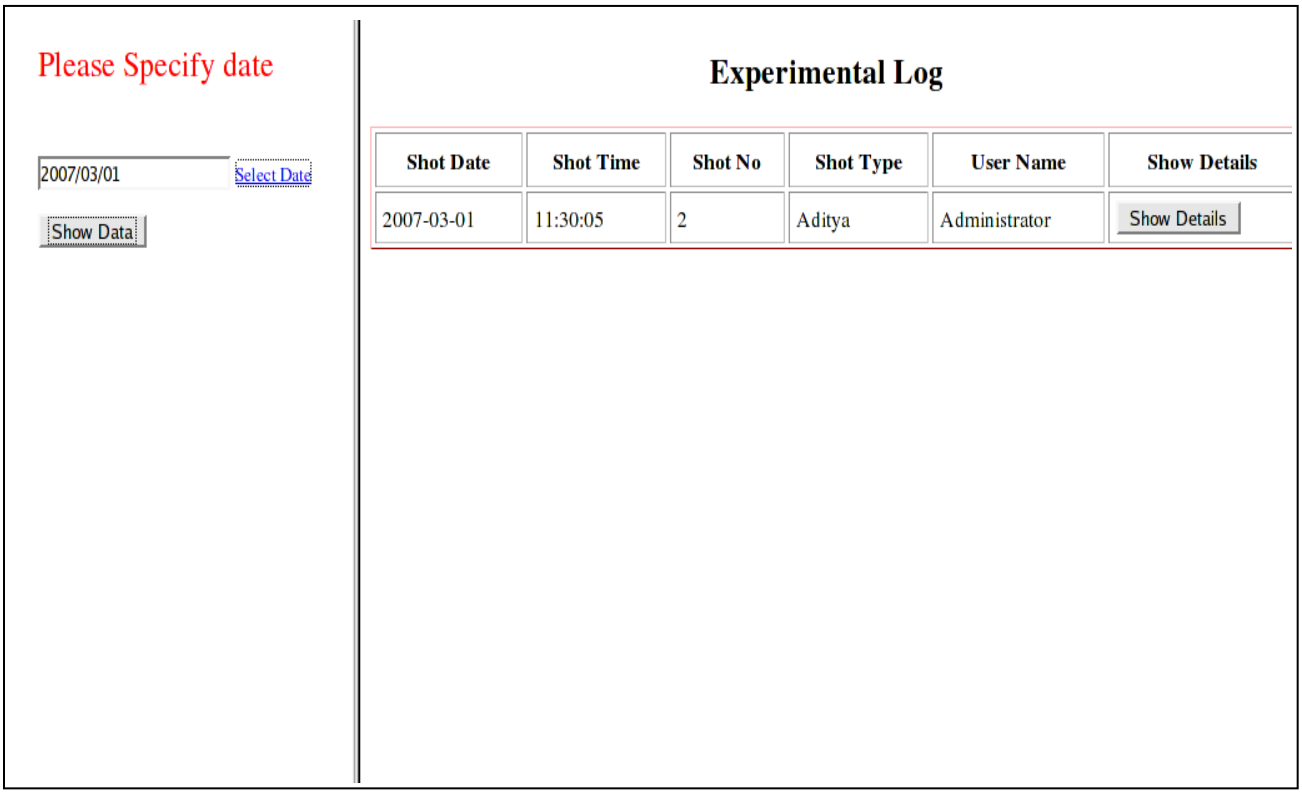


Fig. 5.6: Experiment Analyzer Module Screenshot -1

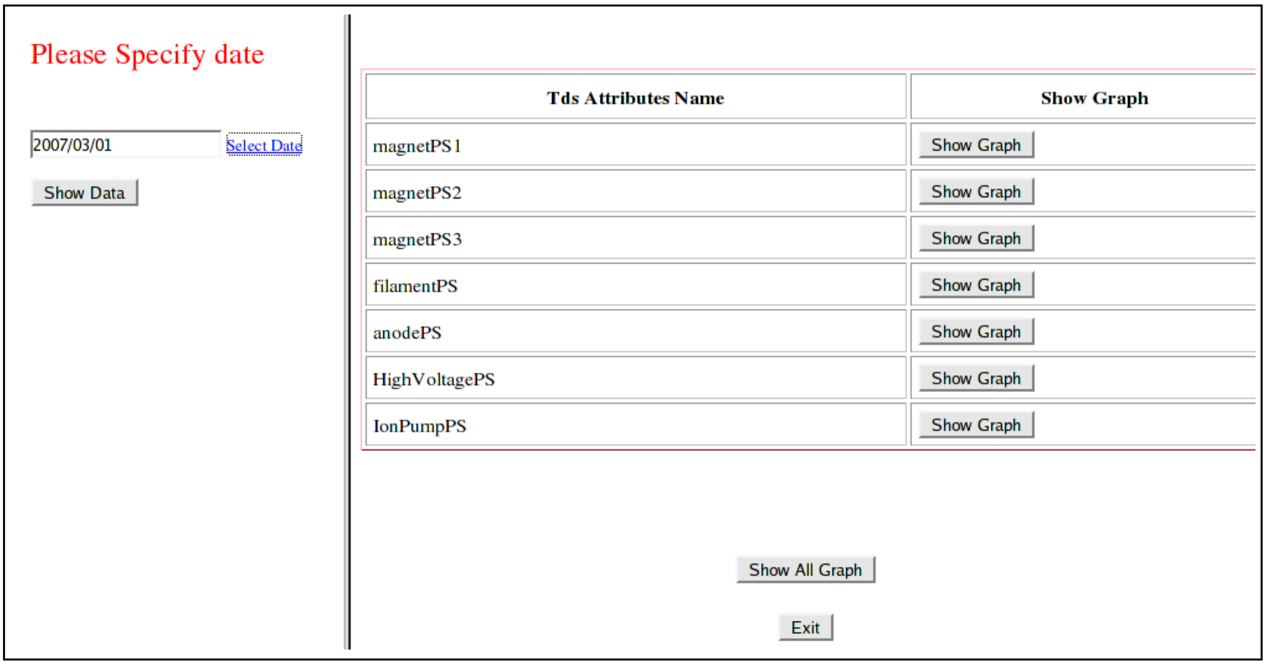


Fig. 5.7: Experiment Analyzer Module Screenshot -2

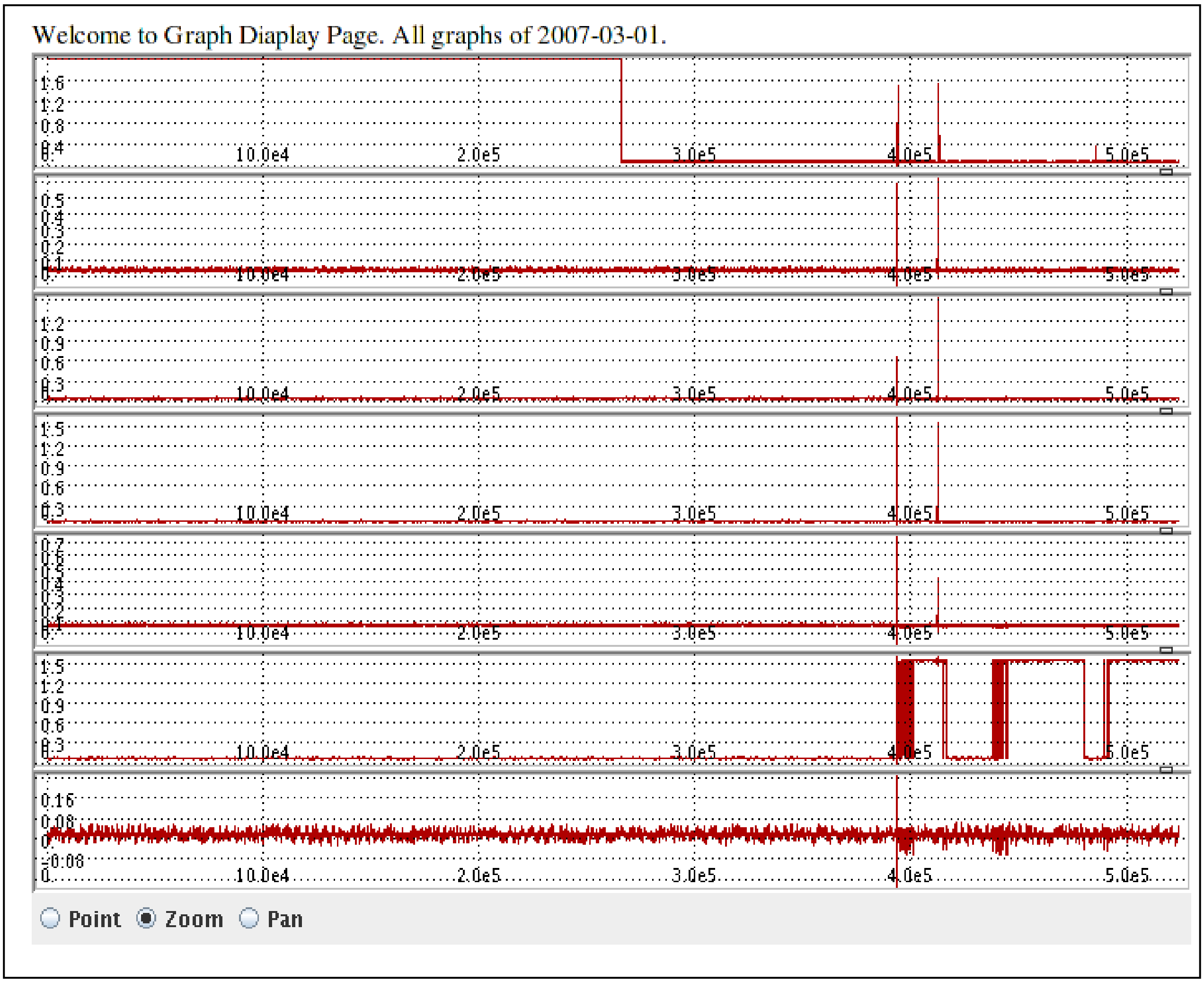


Fig. 5.8: Experiment Analyzer Module Screenshot -3

6. SOAP PERFORMANCE EVALUATION

It is known that the communication in Web Services uses SOAP- Simple Object Access Protocol which is based on XML. Here, an experiment with SOAP in a Web Service Environment is performed to find out the response time using SOAP. It turns out that a direct and naïve use of SOAP would result in a degradation of response time. The major reasons are identified which degrade the performance of SOAP as per the information available and some performance improvement techniques are evaluated. After applying these techniques, the performance becomes better.

6.1 PURPOSE

In this report, performance part of the communication between the provider and the consumer of a Web Service is focused. The performance aspect of the SOAP protocol in a Web Service (WS) environment is evaluated and factors are identified, which have impact on the performance.

6.2 SOAP

SOAP is a lightweight protocol based on XML. In its simplest form, SOAP is one way but may also be used in more complicated transactions like EDI (Electronic Document Interchange) and RPC (Remote Procedure Call), which was one of the design goals when creating SOAP. [9]

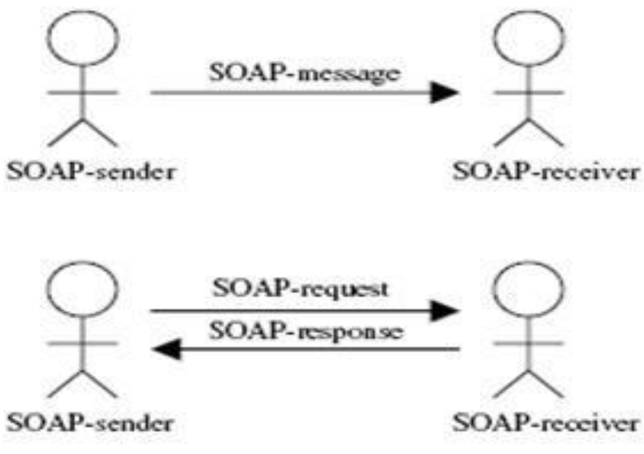


Fig 6.1: SOAP Messaging

6.2.1 SOAP message

The SOAP message consists of SOAP Envelop, SOAP Header which is optional and mandatory SOAP Body field. SOAP Envelope encloses the SOAP message. SOAP Header contains information for the SOAP-node, the processor of the SOAP-message, how to process the SOAP-message. This may be authentication, routing, delivery setting and more. SOAP Body contains information that is targeted to the SOAP-message receiver, for example a service that an application makes requests to. [9]

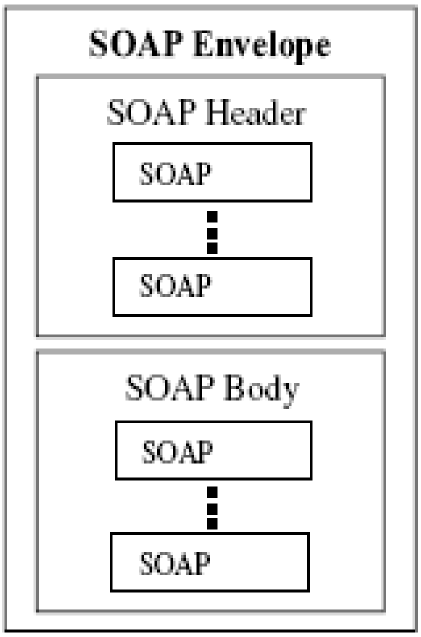


Fig 6.2: SOAP Envelope

6.2.2 Transport of SOAP

The SOAP-envelope is independent of what transport protocol is used. Examples of protocols that may be used are HTTP (Hyper Text Transfer Protocol), SMTP (Simple Mail Transfer Protocol), FTP (File Transfer Protocol) or POP3 (Post Office Protocol). The protocol mostly used today is HTTP over port 80. The main advantage is that the SOAP envelope is integrated into a standard HTTP request and is interpreted as a Web request by firewalls. This removes costly reconfigurations of existing communication management and security policies. One other advantage is that HTTP is a request-response-based protocol, which is compatible with the RPC-paradigm.

6.2.3 XML-parsing

The SOAP messages must be parsed and interpreted before they can be invoked, which is done by the XML parser. There are two types of parsers, Document Object Model (DOM) and Simple API for XML (SAX).

6.2.3.1 DOM

DOM parsers parse the XML document and create an object-oriented hierarchal representation of the document, which can be used at run-time. The structure is easy to use but the technique is very resource demanding both in terms of memory and CPU.

6.2.3.2 SAX

SAX parsers do not store any information. Instead they scan the information and call handler functions that are associated with specific instructions and tags in the XML-document. The SAX parser is often faster than the DOM parser and is not as resource intensive. The drawback is that the logic for handling the XML document instructions and tags is significantly more complex. [11]

6.3. METHOD

The main goal was to find possible improvements that will make the usage of SOAP more palatable. The method to do this was divided into six steps:

1. Implement a prototype that simulates an application in a Web Service environment.
2. Measure the performance with and without load to identify the performance of SOAP.
3. Analyze the results obtained.
4. Identify the causes for the slower performance with the help of available tools.
5. If possible, then find out and apply available improvement techniques for the SOAP implementation.
6. Measure and evaluate the improvements.

The centre of the method is a prototype in a WS environment. Here, it was a "prototype SOA based Remote Experimentation for High Power RF system at 3.7

GHz" where it was possible to plan, execute and analyze the experiment by a remote user. The prototype consisted of a client and a server part. The measurements were done on the client side. The communication between the client and server was done by SOAP. Load was put on the server by using load generators. The generators had different patterns of behavior simulating four different types of load:

Traffic: The generators followed a specific traffic composition where the requests were set to 10 requests per second using SOAP. The lower number of requests was caused by the fact that the Web server hosting the SOAP application was not able to process more requests.

Stress: Both generators started five threads where each of the threads sequentially sent as many requests as possible.

Peak: Both generators sent bursts of requests with maximum speed and the length of 1 second. The bursts were set with an intensity of 1 burst every other second.

None: The load generators were inactive.

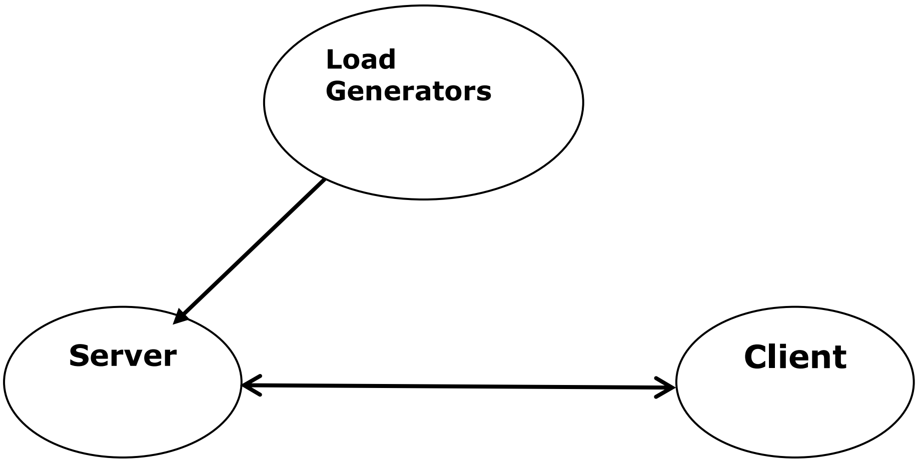


Fig 6.3: Test Setup

The server, client and load generators were placed on different machines. The load generators made requests to the server in the same manner as the client. On the client side SOAP based requests were sent and the response time was measured. This was done during all the different load patterns. The traffic between the client and the server was recorded in order to make a detailed analysis of the packets in the SOAP requests.

6.3.1 Variables

There were two variables that were examined closely when using SOAP. These were:

HTTP 1.0 versus HTTP 1.1

Choice of XML-parser

6.3.2 Experimental Setup

The experimental setup environment consisted of:

1. HP Intel Centrino Duo 1.7GHz with 1GB RAM, used as a server.
2. HCL Desktop Pentium IV 2.3GHz with 1 GB RAM, used as a client.
3. HCL Desktop Pentium IV 2.3GHz with 1 GB RAM used as load generator.
4. HCL Desktop Pentium IV 2.3GHz with 1 Gb RAM, used as load generator.

Ethereal, version 0.99.0, was used as the analyzer. The traffic was captured between the server and client SOAP function calls.

6.4 RESULTS

After performing the experiments, the various results were obtained which can be broadly divided in the following categories:

6.4.1 SOAP Response Time with Different Traffics

The experiment was started with the number of function calls to 5000 in the measuring thread when the load generators were used. In theory the traffic pattern should make fewer requests per second than the stress pattern but in reality the traffic pattern reached the limit of the Sun Java Application Server 9.0 capacity and this caused that the results for stress and traffic to the similar.

Table 6.1: SOAP Response Time Result

Type of Interference	Average Time (milliseconds)
None	338.6
Stress	374.1
Peak	351.5
Traffic	378.6

When the SOAP implementation was stressed the response time was increased by 10%. This implies that there were costs in the SOAP implementation that were independent of the load and this made us focus on other parts than the load cases. Now, the center of attention was changed to the traffic analysis of the packet exchanges when there was no load in order to find discrepancies and causes for these costs.

6.4.2 Parsers

The parsers that were compared were the Crimson and Xerces implementations. The results are shown in table and figure. Predictably SAX parsers were significantly faster than DOM parsers from the same manufacturer. The Xerces SAX parser was on average 2.25 times slower than the Crimson SAX parser as shown in figure 5.4 and in Table 5.2. Even the Crimson DOM parser was faster. Of course this is only applicable for the types of XML documents that we used, which were quite small. [14]

Table 6.2: XML Parser Comparison

Parser	**Average time (milliseconds)**
Xerces DOM	9.872
Xerces SAX	7.5172
Crimson DOM	5.025
Crimson SAX	3.35

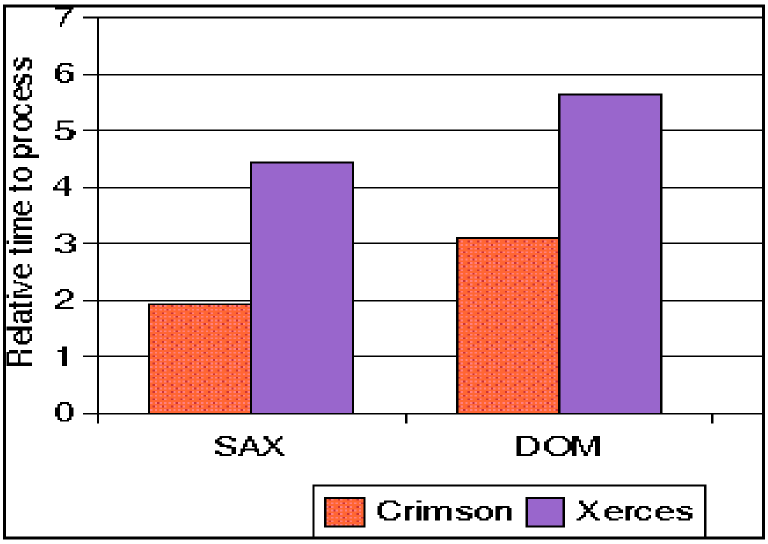


Fig 6.4 Relative time Taken by the Parsers

6.4.3 TCP/IP Traffic Analysis

Putting load on the servers showed that the load itself were not responsible for the large differences in response time for SOAP implementation. The TCP traffic was analyzed as shown in figure 5.6 in detail to find other explanations.

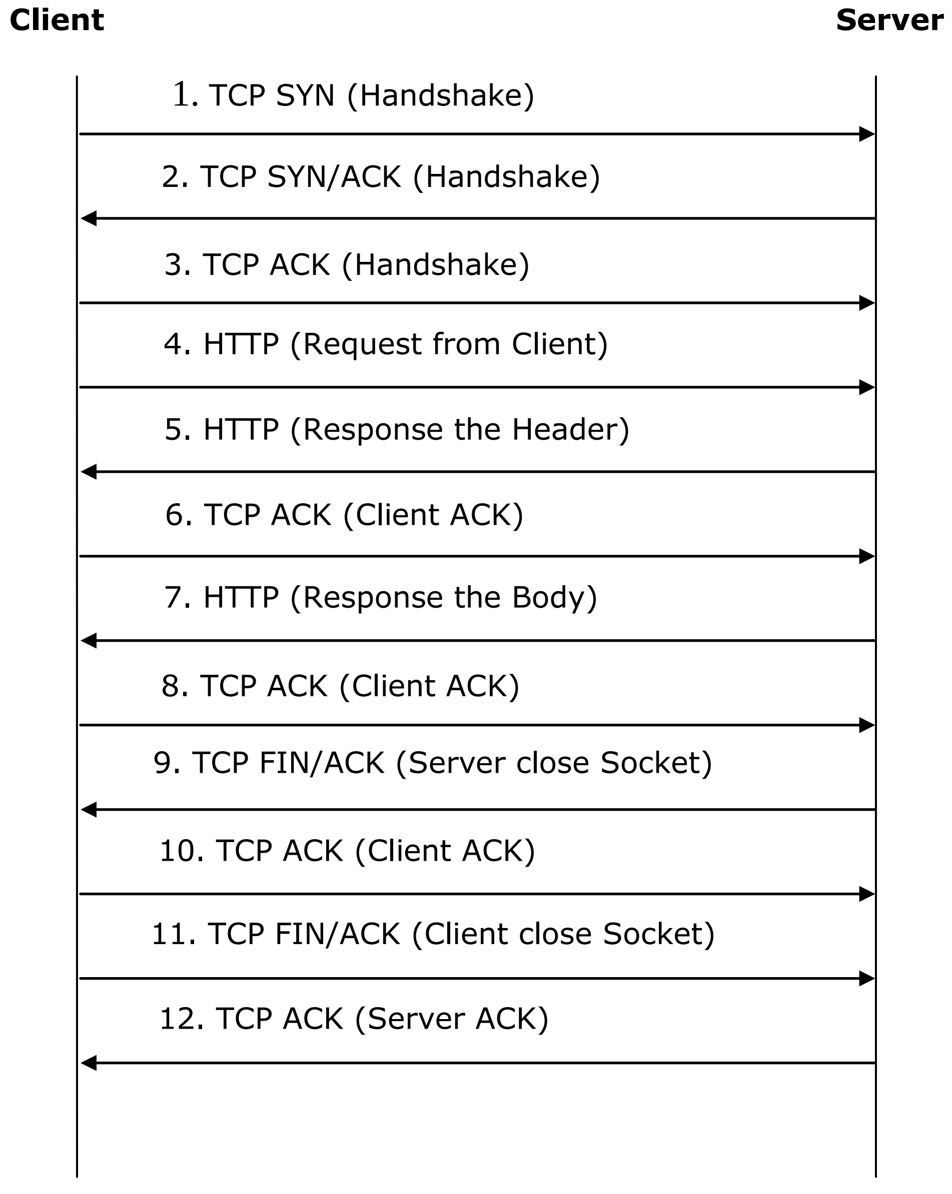


Fig 6.5 Traffic for a SOAP Call

The accumulated average inter-arrival times for each packet in SOAP calls are described in table 6.3. The packets of interest are 5, 6, 8 and 9.

Table 6.3: Average Interval Time for each SOAP Packet

Packet	Average (millisecond)
1. TCP SYN (Handshake)	0
2. TCP SYN/ACK (Handshake)	0.1
3. TCP ACK (Handshake)	0.03
4. HTTP (Request from Client)	1.6
5. HTTP (Response, the header)	18.0
6. TCP ACK (Client ACK)	70.4
7. HTTP (Response, the body)	0.20
8. TCP ACK (Client ACK)	97.6
9. TCP FIN/ACK	171.2
10. TCP ACK (Client ACK)	0.03
11. TCP FIN/ACK (Client Close Connection)	0.1
12. TCP ACK (Server ACK)	0.3
Sum	**359.56**

6.4.3.1 Packets 5, 6 and 8

The HTTP packet (packet 4) from the client contains the request to the server. The server then returns two HTTP packets (packet 5 and 7) where the first packet contains the header and the second packet contains the body of the response which seems odd. The problem was that data sent in the same segment as the HTTP header was ignored. When the client gets the first HTTP packet the client will wait a certain time to see if a new packet in the established TCP connection will be sent so the ACK can get a free ride. This is a feature called **TCP delayed ACK** in the TCP specification and the purpose is to reduce traffic. This delay was measured and the average delay time was 70 milliseconds. This happens again when the second HTTP packet (number 7 in figure 6.5) reaches the client. This delay though is almost constant to 97-100 milliseconds (see Table 6.3). The server will not send packet 7 (HTTP body) before the ack on the previous packet (HTTP header) has arrived. This behavior is caused by the **Nagle algorithm**, [12] which was implemented to stop the sending of repeating small packets.

The conclusion, for these packets, is that they on average make a delay of 170 milliseconds for each SOAP request. The source for these delays is that both the TCP delayed ACK on the client and the Nagle algorithm on the server is active at the same time. If either of them had been inactive the delays would not have an impact on the response time. These delays were tried to remove by improving the client and server implementations.

6.4.3.2 Packet 9

This packet is responsible for the major part of the response time of a SOAP call with an average of 172 milliseconds. This is caused by the implementation of the client. Here, the client side waits until the server side closes the socket.

6.4.4 Improvements

There are a number of possible improvements that were identified:

1. Parse the content length in the first HTTP packet of the response from the server.
2. Send the server response in one HTTP packet.
3. Disable the Nagle algorithm.

6.4.4.1 Parse the content length

The main alternative is to parse the content length field in the HTTP header and stop listening to the socket and close it when specified numbers of bytes are received. This should remove the delay times for packet 8 and 9. The average response time was 328 milliseconds without load generators. If average parsing time (approx. 7.51 milliseconds) is added at the client side very similar values as encountered in our measures (336 milliseconds) can be reached. The TCP traffic in a SOAP call for this implementation is shown in table 6.4. It is observed that when reaching the bold line in the table the client stops listening to the socket and is able to continue execution. The TCP traffic flow for the SOAP call is shown in figure 6.4.

Table 6.4: Average Interval Time for each SOAP packet after parsing content length

Packet	Average (millisecond)
1. TCP SYN (Handshake)	0
2. TCP SYN/ACK (Handshake)	0.14
3. TCP ACK (Handshake)	0.06
4. HTTP (Request from Client)	0.8
5. HTTP (Response, the header)	8.1
6. TCP ACK (Client ACK)	86.7
7. HTTP (Response, the body)	0.12
Sum	**95.92**
8. TCP FIN/ACK (Client close socket)	0.78
9. TCP ACK (Server ACK)	0.1
10. TCP FIN/ACK (Server Close Connection)	72.0
11. TCP ACK (Client ACK)	0.06
Sum	**168.86**

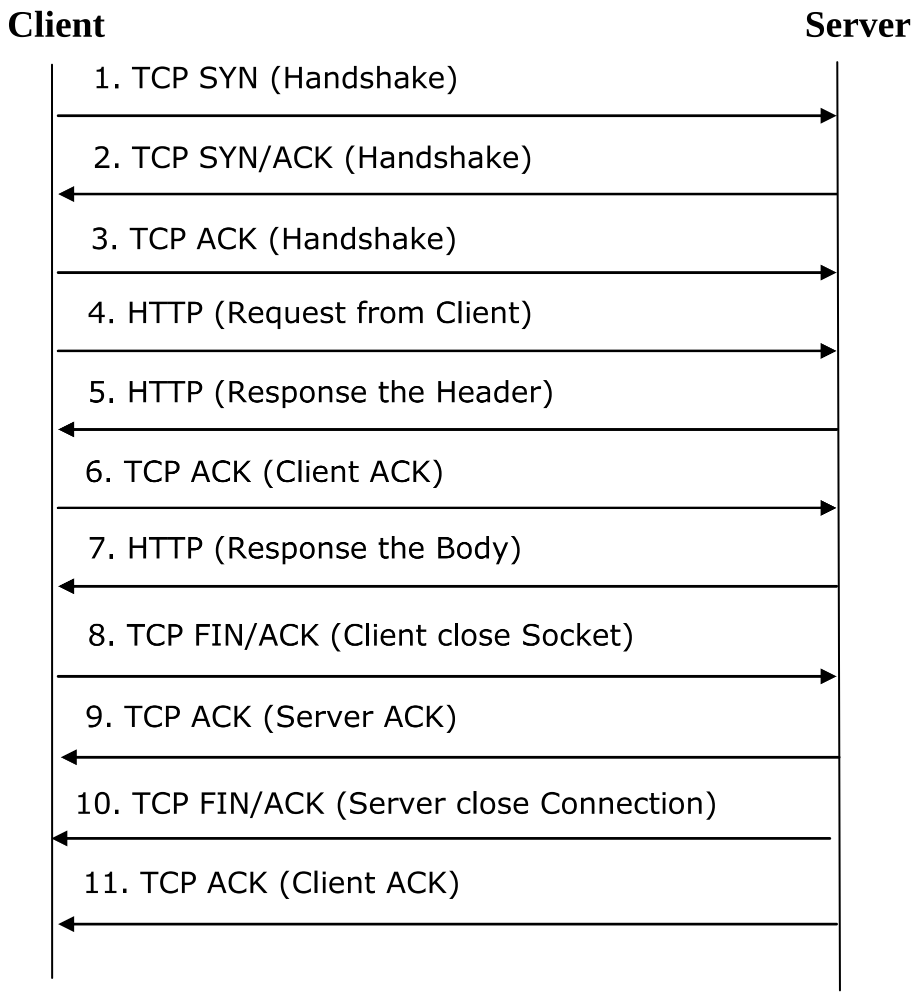


Fig 6.6 Traffic for a SOAP Call in case of Client Closes Socket

6.4.4.2 Response in one HTTP packet

It is desired that Web server which received the request from the client should return response in one HTTP packet. The result for this implementation will imply that packet 6 will removed and the delay of on average 70 milliseconds will be eliminated. This case can not be addressed in this project.

6.4.4.3 Disable the Nagle algorithm

It is possible to disable the Nagle algorithm in Java when using sockets. This should be done on the server side. The tcp_nodelay parameter for sockets is set to true and the delay for Nagle algorithm is set to zero.

6.5 DISCUSSION

The inter arrival times for each packet in a SOAP call is shown in this chapter. It can be viewed that packets 5, 6, 8 and 9 stands for the major part of the SOAP request response time. The delays are caused by an unfortunate combination of the TCP delayed ACK at the client and the Nagle algorithm at the server.

In short, the summary of this chapter is that the implementation of the communication handling at the server and the client is of high importance. In the initial test setup the response time was on average 338 milliseconds including server and client execution time. With improvements that were realizable to make in the prototype, the response time was lowered to 96 milliseconds and execution time on server and client was the same. In real Web Services the execution time on the server and the roundtrip time will probably increase.

7. CONCLUSION

7.1 CONCLUSION

The aim of the project was to perform remote experimentation for High Power RF Test System at 3.7 GHz using Service Oriented Architecture and to improve the performance part of the communication between the provider and the consumer of the web service.

To achieve the target, Experiment Planner, Experiment Executor and Experiment Analyzer modules are developed. To check functional behavior, each module is tested individually and finally all these modules are integrated well into an entire system unit and tested successfully at real VME based Data Acquisition and Control Setup at RF lab. Using these modules, remote experimentation is performed successfully. The system is flexible and modular to be enhanced for lower hybrid current drive system operation in future.

During the SOAP performance evaluation, initially, the response time was on average 338 milliseconds including server and client execution time. With improvements by analyzing the traffic for a SOAP call, the response time was lowered to 96 milliseconds. However execution time on server and client remains same. The response time is reduced by nearly 350% i.e. nearly less than one third.

7.2 FUTURE WORK

High Power RF Test System at 3.7 GHz is compromised of a number of heterogeneous and distributed components. This project is just a prototype that addresses only one klystron operation. The same base can be extended to bring all the LHCD system operation into one umbrella.

The database is used to keep the log and records of the experiments. For long duration operation of the LH-DAC, the knowledge filtering and representation will be an issue of study. The response time of data representation will be an area of interest.

After doing performance analysis and making improvement in the SOAP communication, there are still some issues that have to be carried out for the enhancement which are mentioned as below:

impact on the performance of response time of different sizes of the SOAP messages

Performance testing in different network environment for example over the internet to find the real communication cost and

impact on the performance of security addition to the Web Service

REFERENCES

1. D. Bora , S. Dani, S. Gangopadhyay, B. Jadav, M. Jha, B.R. Kadia, P.L. Khilar, S.V. Kulkarni, M. Kushwah, A.P. Patel, K.G. Parmar, K.M. Parmar, P. Parmar, K. Rajnish, S. Raghuraj, K. Sathyanarayana, P. Shah, P.K. Sharma, Y.S.S. Srinivas, R.G. Trivedi, "Steady state high power test at 3.7 GHz", Fusion Engineering & Design, Elsevier Publication, page no. 56-57,2001
2. D. Bora, P.K. Sharma, S.L. Rao, R.G. Trivedi, M.K. Gupta, K. Ambulkar, K. Mishra, P.R. Parmar, K. Kumar, A. Thakur, " Design criteria and current status of LHCD system on SST1 machine", Fusion Engineering and Design , Elsevier Publication , page no. 141–152, 2007
3. Yih-Cheng Lee, Chi-Ming Ma Shih-Chien Chou, "A Service-Oriented Architecture for Design and Development of Middleware",14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE'05), 2005
4. Booth, D., et. al. (editors), "W3C Working Group Note 11: Web Services Architecture", World Wide Web Consortium (W3C), February 2004, http://www.w3.org/TR/ws-arch/#stakeholder
5. James McGovern, Sameer Tyagi, Michael E. Stevens, Sunil Mathew, Java Web Services Architecture, 1st Edition, Elsevir Publication, January 2005
6. Liam O'Brian, Len Bass, Paulo Merson, "Quality Attributes and service Oriented Architectures", Software Architecture Technology Initative, September 2005, Technical Note CMU/SEI-2005-TN-014
7. Sanddep Chatterjee, "An Introductory Overview to Web Services", CSI Communication Journal, page 6-11, March 2006
8. Jim Webber J., "An Architect's Guide to Developing Enterprise Web Services", CSI Communication Journal, pages 15-19, March 2006
9. SOAP Version 1.2 Part 1: Messaging Framework. http://www.w3.org/TR/soap12-part1/
10. SOAP Version 1.2 Part 0:Primer. http://www.w3.org/TR/soap12-part0/
11. The official website for SAX Parsers. http://www.saxproject.org/
12. Nagle, J. Congestion Control in IP/TCP Internetworks. RFC 896. January 1984.

13. Robert Elfwing, Ulf Paulsson, Lars Lundberg, "Performance of SOAP in Web Service Environment compared to CORBA", Proceedings of the Ninth Asia-Pacific Software Engineering Conference (APSEC'02),2002
14. Violleau, T. Java™ Technology and XML Part 2: API Benchmarks. 2002. http://developer.java.sun.com/developer/technicalArticles/xml/JavaTechandXML_part2/
15. Davis, D. Parashar, "Latency Performance of SOAP Implementations", IEEE Cluster Computing and The Grid 2002.

APPENDIX-A

NETBEANS IDE 5.5

Salient Features:

Netbeans IDE 5.5 provides SOA tools through which Web Services can be easily created. It provides additional service editing, orchestration support, XML development and secure web service development support.

Netbeans IDE 5.5 helps in Web Service creation through WSDL editor for rich design time support and supports both the Java EE 5 and J2EE 1.4 Web Service architectures. It also provides the facility of wizard based Web Service creation.

It provides Web service orchestration through BPEL mapper which visually author BPEL 2.0 business processes with the BPEL designer and deploy to the built in BPEL engine.

It supports XML schema development with the XML schema editor, which helps in schema based code completion for XML instances.

Netbeans IDE deals with the complexity since there are many standards and protocols and IDE takes care most of the plumbing codes. Basically it helps in focusing on the business domain rather than being puzzled in the syntaxes and various programming language constructs. It helps in a great manner to visualize, analyze and debug the applications. So in this way, helps to increase the productivity and quality of the work.

APPENDIX-B

SUN JAVA APPLICATION SERVER PLATFORM EDITION 9.0

Salient Features:

It provides the foundation for delivering enterprise-class application and Web services.

It offers a unique modular architecture based on some of the industry's most proven, high-performance, and standards-compliant components.

It offers tools that help in deploying applications quickly.

It is completely free of license fee for development, deployment, and redistribution.

Unlike other commercially available application servers, the Application Server PE 9.0 is built on the very platform that defines the Java EE standard—Java EE 5 Reference Implementation source code. This assures the most rigorous Java EE standards compliance and Web services interoperability through support of the Web Services Interoperability Profiles (WS-I). Enterprises and application vendors benefit from this Java EE standards- focused approach, because it reduces the risk of proprietary lock-in and enables compliant applications that are portable across compliant application servers without costly modifications.

It provides additional features such as higher performance and load balancing, as well as clustering and monitoring capabilities. Optimized for Web services, the Java System Application Server is designed to help enterprises and service providers maximize their freedom of choice through Java technology's "Write Once, Run Anywhere" simplicity.

The Java System Application Server helps enterprises and service providers lower total cost of ownership, accelerate time to market, and increase productivity.

Foundation for Service Oriented Architecture (SOA)

The Java System Application Server PE 9.0 is an ideal platform to develop SOA applications. Application Server PE 9.0 is integrated with the Java Web Services Developer Pack (WSDP) 2.0, which supports the latest Web services standards to make implementing an SOA easier. These standards include the Java API for XML-based Web Services (JAX-WS 2.0), which specifies Web services APIs for the Java platform, and Java Architecture for XML Binding (JAXB) 2.0, which specifies Java and XML binding. With JAX-WS 2.0, the amount of code developers need to write can be significantly reduced.

Application Server PE 9.0 can connect to, and work with, a number of Web services registries. These registries allow an enterprise to track and manage large, increasing numbers of Web services.

Web Services Management

Web services are first-class manageable objects in Java System Application Server PE 9.0. This means that Web services deployed to it are automatically discovered and can be managed and monitored. If monitoring is enabled for a Web services endpoint, information about response times, throughput, requests, and faults is collected and can be viewed through the administration GUI. SOAP message content can also be examined. Meanwhile, a Web services testing page can be automatically generated — eliminating the need for explicit Web services client development.

Performance

In addition to startup, memory footprint, and deployment performance improvements, Java System Application Server PE 9.0 increases runtime performance. It includes a highly scalable HTTP connection handler that is implemented with lower-level Java Native Input Output (NIO) primitives, and can handle thousands of connections with a small number of threads. Depending on the XML message size, processing time is three to five times faster, and message size is 1.3 to five times smaller.

APPENDIX-C

INTRODUCTION TO MDSPLUS

MDSplus 1.0

MDSplus is a set of software tools for data acquisition and storage and a methodology for management of complex scientific data.

MDSplus allows all data from an experiment or simulation code to be stored into a single, self-descriptive, hierarchical structure. The system was designed to enable users to easily construct complete and coherent data sets.

The MDSplus programming interface contains only a few basic commands, simplifying data access even into complex structures. Using the client/server model, data at remote sites can be read or written without file transfers. MDSplus includes x-windows and java tools for viewing data or for modifying or viewing the underlying structures.

MDSplus was designed to give researchers the ability to produce complete, coherent, self-descriptive data sets and to provide tools for efficient access to that data in a distributed computing environment. The result was a system with sufficient flexibility and extensibility that ALL information associated with an experiment (or simulation) can be stored in the same structure and accessed through the same set of simple calls. By unifying setup, calibration, geometry and so forth with task descriptions, scheduling, status and all raw and processed data, MDSplus makes it easy to share data and tools across applications, facilitating collaborative research and reducing duplication of efforts.

Comparison with Other Data Standards

MDSplus is not simply a data standard as it provides support for data acqusition, task description, scheduling and dispatching, asynchronous event generation and distribution, etc. However, MDSplus is a highly capable data storage and management system with a unique combination of features and functions including:

A rich set of simple and composite data types
Hierarchical structure
Self-descriptive - "metadata" included with each data item
Expression evaluation, enabling "indirect" reference to data
Data compression and decompression
Remote access via client/server model
Service vs file oriented - "logical" rather than "physical" data access
Extensible
Scalable to very large and complex data sets

Data Management

Data is stored in a user-defined hierarchical structure
All data is available through the same interface
A built-in expression evaluator extends the capabilities of MDSplus
The MDSplus data structure is self descriptive
MDSplus data is available remotely via the client/server model

APPENDIX-D

USER GUIDE

A detailed description of each screenshot of chapter 5 is given here to understand how a remote user interacts with the system. The description associated with each screenshot explains the functionality.

ROLE BASED AUTHENTICATION

The screenshot shown as in Figure 5.1 shows the authentication procedure. This web page is displayed to every user whatever its role is defined. A remote user supplies the login information which includes the user identification (userid), password and the role. As mentioned in chapter 2, four different roles are defined for planner, executor, analyzer and system administrator respectively. After filling the required information, the supplied inputs are checked against stored information in the database. A table is maintained in the RDBMS which compares all the inputs. In this way, it authenticates the remote user only if all the given information matches against the stored values otherwise the request for login is rejected. If permission is granted, then next module is displayed to the user depending on the role selected by the user.

EXPERIMENTAL PLANNER MODULE

When remote user selects the role as Experiment Planner, planner module is displayed to user as shown in Figure 5.2. An experiment planner first selects the purpose of the experiment among available options. Then the various input parameters like Magnet Power Supplies, Filament PS etc are chosen from the list available. The experiment planner chooses values within a particular range provided, based on the previous experiment results and available in literature. When the planner presses show result button, a theoretical simulated output is shown as shown in Figure 5.2 which consists some of the output parameters. After being satisfied with the output, planner can save the supplied input data into log. An alert verifies that the data is saved successfully. Now, a planner can set the test cases against this particular experimental plan. Clicking the set test plan button, displays the screen as shown in Figure 5.3. Now the planner supplies the various parameters regarding to test cases. Test cases can be dynamically added and deleted at run time. Finally, the user saves all these test cases into the log which will be made available at the Experiment Execution module. After closing the window of test cases, planner exits form module.

EXPERIMENTAL EXECUTOR MODULE

When remote user selects the role as Experiment Executor, Experiment Execution module is displayed to user as shown in Figure 5.4. First a blank screen appears to the user containing only one button labeled as 'show objective'. After pressing the button, the recent planned experiment with the associated test cases is shown to the executor in the left most and in bottom part of the screen respectively. The test cases can be navigated with the help of next and previous buttons. When the user presses the start operation button, VME session gets started by showing green indicator at the upper most part. Now the screen also contains the component status, current value of characteristics signals and the control panel. The component status show the current status of components that whether it is in ON or READY or TRIP state. The green indicator shows on status while red indicator shows off. Present value of current and voltage is shown through in characteristics signal screen. Initially, status of all components shows off state and characteristics signals are disabled. When executor sets the current or voltage through the control panel, a command is run on the VME server and it is shown through the green indicator. The component comes in ON state and after achieving a particular value, it comes in REDAY state. Any fault causes the component to come in TRIP state. The components are booted in a particular sequence which is known to the executor in prior. After setting the values for all components, any particular test case is operated. A successful test case generates output and it is shown in the output area upper to test case as shown n Figure 5.5.

EXPERIMENTAL ANALYZER MODULE

When remote user selects the role as Experiment Analyzer, Data Visualization module is displayed to user as shown in Figure 5.6. An experiment analyzer views the shot details by selecting the particular date from a popup window where it can navigate through the months and years. The shot details are displayed in a tabular format showing the shot number and user name. The further details can be obtained by pressing the 'Show Details' button. The activities details as well as Time Dimensional Signal (TDS) attributes are shown as a result in Figure 5.7. Now the user can view individual graph of each component as well as graph of al component in a single window. The main advantage of it that t provided the facility for comparison for better analytical result. The window of all graphs is shown in Figure 5.8. The same functionality is obtained as a user gets from jScope. The user can perform zoom, pan or point in the graph.