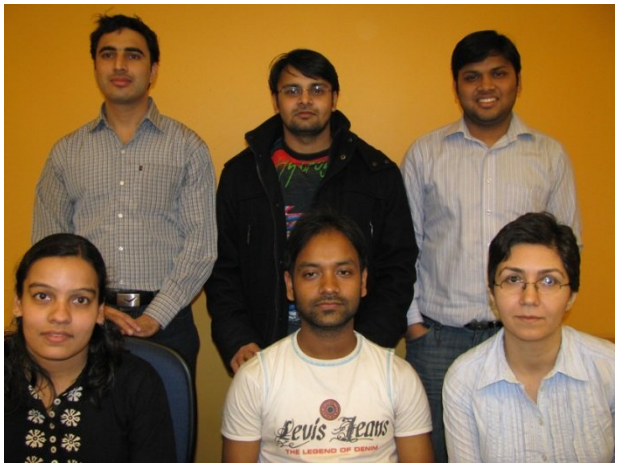
DPS - Distributed Polling System

Project Members:

Jenis Kavadiya, Avijit Dutta, Rishabh Gupta, Aparna Vijaya,
Sajjad Ali Khan, Farahnaz Yekeh

(jka08004, ada08001, rga08001, ava08001, skn08008, fyh08001)
@student.mdh.se



http://www.fer.hr/rasip/dsd/projects/distributed_decision

Mälardalen University, School of Innovation, Design and Engineering
&
University of Zagreb, Faculty of Electrical Engineering and Computing

Table of Contents

1. Executive Summary

Distributed Polling System is a distributed software system that facilitates important business decisions to be taken without having any conference call or adopting any other media of conversation, where decision makers are scattered across the world. DPS allows decision makers to compose a business issue, to intimate the members present anywhere in the world and to collect voting responses, through SMS and Email. In addition, the DPS architecture supports further medium of intimation like voice mail or video stream.

In project start-up, we faced few ambiguities in our understanding of the requirements and we had to start successive phases like design, implementation and so on at early stage in parallel to requirements engineering due to time limitation. However we started with the portion of the requirements that we had pretty much clarity and started designing the architecture along with a little bit of implementation. We used waterfall model for project planning purposes and followed Scrum approach because of the small team size, more reliance on informal communication and less requirements clarity in inception phase.

From SCORE perspective DPS focuses more on architecture, design, algorithms and interfaces, which are the most critical parts for evaluation. Considering the same we have revealed a robust architecture design that facilitates scalable, technology independent, loosely coupled and component based system.

To fulfill, we designed DPS as a software system that consists of software applications. SMS Gateway, web-DPS application and Email server are the software applications in distributed polling system aka DPS. All the software applications in DPS are inter-connected through JBOSS middleware which introduces high level of loose coupling. In the system middleware is in the hub and each application is situated on the perimeter of a bicycle-wheel linked through spoke called ‘adapters’ in software terminology.

Time limitation, complexity of the system and adoption of such architecture introduces a number of independent components whose integration is a challenging task

DPS is packaged software that can be installed and further configuration will ensure communication among its software applications.

A handful of technological challenges have been faced during the implementation phase. Major challenges include understanding telecom network, Integration of Software applications using middleware - the core of our system, software interaction with mobile handset and Unavailability of university Email server.

Resolution includes hard work of team members to grasp telecom network functionalities and to expertise in JBOSS middleware product suite, implementation of java service to interact with the mobile handset through COM port and successful installation, configuration of Microsoft WINMAIL server version4.6 respectively.

However, our confidence has been boost-up after successful integration of this huge number of components that has produced system’s expected functionalities through proper testing using well formulated test design techniques as per our architecture design.

Our project team is versatile in nature from various perspectives like cultural differences, work experience, preferences of time in project work and so on. Team consists of six members from India, Iran and Pakistan. We have achieved our success by investing twenty hours per person per week for almost ten weeks, due to having few other courses in parallel in the same curriculum.

2. Introduction

The goal of the project is to build a system as Distributed Polling system (DPS) to vote on a certain topic, capture their votes, calculate votes and announce result through technological infrastructure. SMS and email are the medium to be used for intimation and voting. DPS provides the solution with scalable, technology independent and loosely coupled architecture to facilitate smooth and convenient business decisions.

We are the international students continuing with this project because of distributed software development academic curriculum. Our project team consists of six international students and developing this software with around ten week's time-line.

The report is organized as follows.
In initial sections, we discuss *scope of the project and main challenges, project requirements* and *architectural design*.
In later *section implementation, Verification and validation, development process, project plan, management plan* and *outcomes and lessons learnt* are discussed. Finally we have summary and references sections.

"Scope of the project and main challenges section" depicts DPS project scope along with serious challenges we were supposed to face during project start-up.
"Project requirements" section analyzes the problem statement of our project and requirements specification. This also discusses the ambiguous requirements we faced and their resolution after discussion with stakeholders.
"Architectural design" section delineates how we formulate our project requirement into a scalable and technology independent architecture. This also depicts how the introduction of middleware provides utmost loose coupling to enterprise applications.
In "implementation" section we depict how our development is following the architecture thoroughly. Even this section specifies how the architecture emphasizes to create granular components which in-turn leads to develop, integrate and test the system requirement.
"Verification and validation" section specifies different test design techniques used and test cases derived from that to ensure system's functionality as per the requirements.
"Project plan" specifies our planning activity using waterfall model and "management plan" dictates managerial decisions taken for timely project completion.
Finally it is the "outcomes and lessons learnt" section that delineates best practices and our experiences from this project implementation.

3. Scope of the Project and Main Challenges

DPS is part of distributed software development course which runs for a duration of ten weeks as a part of Masters in software engineering program in Mälardalen University, Sweden and University of Zagreb, Croatia. The aim of this course is to enable students gaining experience in distributed development environment. The project team consists of students from both Sweden and Croatia in order to achieve distributed environment. Participation in SCORE and its result is one of the major evaluation criteria of this course. Both universities have international students which makes project team representatives from various countries. This provides ideal environment for a distributed project in terms of geography and different cultures.

The major challenges we have identified in the whole process were:

- Team members are from different countries having different work cultures which may lead to misunderstandings during the development process.
- Communication among team members is another main challenge because of distributed environment and also many team members does not have English as a mother tongue which can cause problems during intercommunication among team members.
- The contact with customer (SCORE supervisor in this case) is limited and through emails only. This is a major challenge for requirement engineering phase. There are many

ambiguous requirements we have already identified in the functional requirement document provided by SCORE and clarification of these requirements are time consuming as mode of communication is only email.

- The actual time we have for completing the whole project is ten weeks. This leads careful selection of development process because time constraint is one of the major factors while deciding upon development process in any project.
- Another important challenge is selection of technology as different team members are skilled in different technologies and are good in different phases of development cycle. It is really important to create and assign roles based on skills we already posses in order to produce most productive and efficient outcomes.

4. Requirements

4.1 Problem Statement

DPS is a software system which can be used to intimate, vote and obtain voting results for a business decision for a predefined set of members or business users. The users of the system need not to be connected to internet or to be in some workplace in order to use the system. System should allow them to vote using their mobile phones (using SMS, email and voicemail so on).

The major functionalities of DPS are the following:
DSP should:

- Have a web based interface to manage decision information, group member's information and other administrative tasks
- Send a message to inform about a new decision which has to be taken to predefined set of users or group of users
- Receive the voting information from the members and take care of other issues such as security measures, duplicity of messages etc
- Calculate and announce the result to all members of the voting group.
- Allow sharing of views among group members.

4.2 Requirements Specification

Following the main requirements identified in the Problem Statement we classify them in the functional and non-functional requirements. They are discussed below.

4.2.1 Functional Requirements

General requirements. The requirement of DPS is to develop an application which enables polling among predefined set of members in order to make some specific business decision. The application can be hosted on a web server and users of the system, i.e. poll members and system administrator can use the system in order to perform various functionalities explained in sections discussed below. The poll members can cast votes by sending SMS or email based on their current availability, for example, if one or more poll members are traveling then they can use their mobiles to send SMS to send their vote. DPS will be able to send information of polls, receive member's responses and announce result of polls based on the responses received. Later on, these results can be used by management to take important business decisions. The main actors of DPS are system administrator and poll member.

SMS Module Interaction Requirements. Using mobile communication for voting on a certain issue in order to make business decision is one of the goals of this system. Sending SMS text messages is a way for communication that makes it possible to vote between predefined options and receive voting results. Sending alert message to members in the correct format through SMS, receiving the acknowledgment after the data is received and sending the analyzed results of each vote are requirements which will be done by SMS module Interaction.

Email Module Interaction Requirements. Another type of mobile communication is sending and receiving information by email. One of the requirements of DPS is Email Module Interaction part, which will do all controlling on sending and receiving email. The emails about decisions regarding

a certain matter will be sent to all members of the group which is pre defined by administrator and their responses will received by system. Controlling the email interaction is an important part of the system which will be responsible for sending alerts to all specified members and getting their results in a secure communication channel with encrypted data.

Database Requirements. Database will store information of members of the system, all information of defined pools, answers of the members and the result of each poll. It must have a good performance with a well designed structure to avoid redundancy. Accessing the database will be possible from data access layer and proper stored procedures will be written for getting data from it.

4.2.2 Non Functional Requirements

Security Requirements. Security requirements allow members to vote and discuss about important business decisions in a secure manner. The data send over the internet by email or by SMS will be encrypted. Also, the detail information related to business decision will be sent to member's email address in the form of password protected pdf file. While receiving responses, system will check whether the sending person is a legitimate user or not. Authentication is achieved using digital signatures.

User Interface. The system is a web-based application. User and administrator can have access to the system from anywhere in the world using the web browsers.

Usability. Clearly show all the options a user or administrator has at any given stage while using the system. We should make the system web page front-end simple and easy to understand and to interact with. It should be easy and painless for the user to create a poll and thereby facilitate easy decision making. If it's too hard or takes too long the user might leave without using this option for online decision making.

4.3 Requirement Engineering

We first identified all combinations of possible scenarios, after that we specified the requirements using use case diagrams and in the document we present one as an illustration.

4.3.1 Matrix for scenarios

By studying the problem statement and description, we have identified several possible scenarios or cases which we may encounter (more than 100 combinations). All the possible scenarios and examples have been listed out in the form of a matrix as well as text so as to get a more clear idea of what should be exactly developed. A sample matrix is depicted as in the following section.
The priority of the poll decides whether the poll can be send by SMS or email or both(for egg: a low priority poll is send only as an email, whereas a high priority poll are send by moth means). Anonymous field represents whether user can vote anonymously or not. It is decided by the poll member during the creation of poll. Discussion allowed specifies whether discussion facility is permitted for a given poll. There are different poll types that we have identified. The poll can be a yes/no type or choosing anyone option from then given list or choosing the options based on priority and choosing many options without any priority. Result Calculation represents when the result can be calculated for a particular poll. Some combinations of poll cannot exist (for e.g.: a priority poll cannot be send only as an email). This is what is specified as Does this type of poll exists.

Table 4-1: Matrix of scenarios

Priority of poll	Anonymous	Discussion allowed	Send by SMS	Send by Email	**Poll type**				Result calculation (final decision)	Does this type of poll exist?
					Yes/ No	Choose anyone option	Choose many options	Choose according to priority		

Low	✔	✔		✔	✔				>50%vote	Yes
Medium	✔	✔	✔		✔				>50%vote	Yes
High	✔	✔	✔	✔	✔				>50%vote	Yes

From the table we arrived at several scenarios like:

1. In a high priority poll where discussion was allowed, one of the poll members sends his opinion. But later when he discussed with other members of poll, he wanted to change his opinion before the poll was closed.
Is this allowed? How is it going to be implemented?

2. There is a case where at the time of calculating results of a poll, there is 50%-50% vote between two options. Is the poll reopened again for voting/discussions? How the system will implement this?

4.3.2 Use Case Model

As shown in Figure 4-1, DPS has the following two actors:

- Poll member provides the system with information needed for creating a poll and thereby decision making.
- Administrator helps to manage accounts of users and poll details.

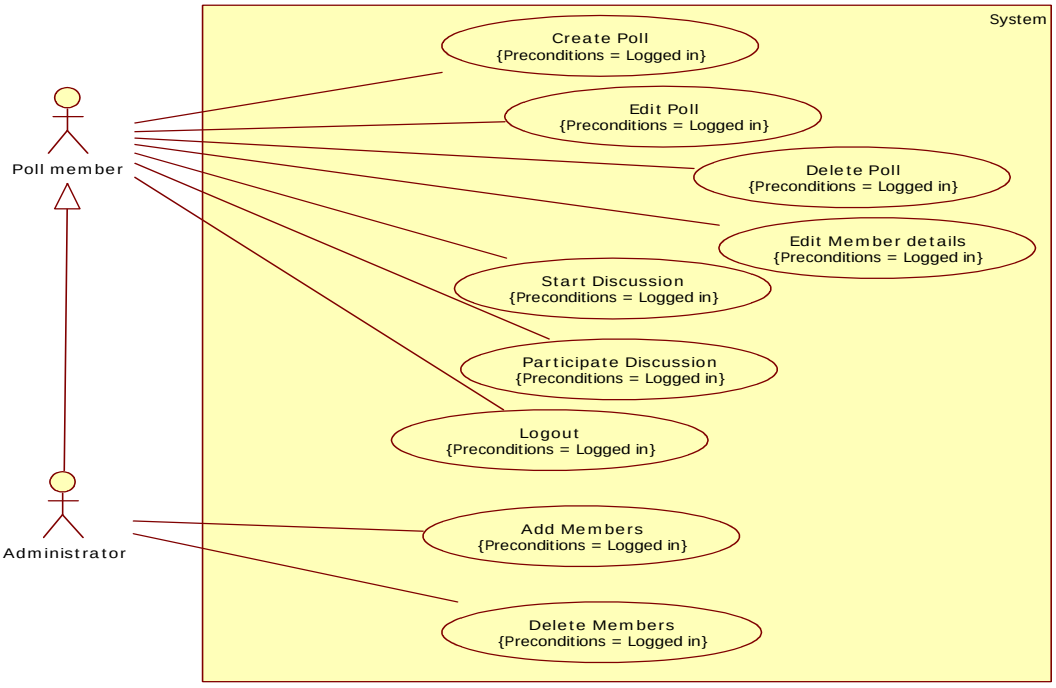


Figure 4-1: **Use case Model**

An example for the use case description is shown as below:

Table 4-2: Use case Description

Use Case Element	Description
Use Case ID	1
Use Case Name	System Login
Use Case Description	A system user can login into the system to perform associated actions. Poll member can create a new poll or view the poll details by logging into the system. Administrator has the privilege to adding and deleting poll

	members, but he also can create polls just like the other members
Primary Actor	Poll member and Administrator
Precondition	None
Basic Flow	User will login into the system
Alternate flow	An error message should be thrown if username/password is incorrect

4.4 Requirement Definitions

We have analyzed the requirements and categorized them as the following requirement groups.

Table 4-3: Requirement groups

Identification	Requirement Group
MAC	Main Application Core
SA	Security Aspects
PY	Persistency
SMI	SMS Module Interaction
EMI	Email Module Interaction
DB	Database
WA	Web Application

Here we categorize all the requirements of the system based on requirement group and priority and are listed in the following table. The keyword Ctm specifies the "Customer" and Sys stands for "Required as a consequence of system".

Table 4-4: List of requirements

Identity	Priority	Description	Source
		Main Application Core	
MAC-1	1	Call for voting on a certain issue	Ctm
MAC-2	1	Inform all members to vote for a business decision	Ctm
MAC-2	1	Broadcast newly created poll to all poll members	Ctm
MAC-3	1	Allow mobile communication by Email or SMS	Ctm
MAC-4	1	Calculate result based on responses received	Ctm
MAC-5	1	Broadcast the result back to all poll members	Ctm
MAC-6	1	Consider Security and privacy measures to avoid duplicates	Ctm
MAC-7	1	Close/Publish the result of vote whenever the result of a vote can be calculated according to the number of votes, percentages of answers or time	Ctm
		Web Application	
WA-1	1	User can log in to the Web page	Sys
WA-2	1	User can manage his/her personal information	Sys
WA-3	1	User can see all open/close polls information	Ctm
WA-4	1	User can create a poll	Ctm
WA-5	1	User can delete the poll which he/she has created	Ctm
WA-6	1	User can select members of each poll while is creating a poll	Ctm
WA-7	1	User can specify all attributes of a poll	Sys
WA-8	2	User can create/participate in a discussion	Ctm
WA-9	1	User is allowed to create Yes/No votes, vote between N (N>=2) options, multiple votes or priority votes	Ctm
WA-10	1	User can create anonymous or not anonymous votes	Ctm
WA-11	1	User can extend the deadline of the poll that is created by himself/herself	Sys
		SMS Module Interaction	
SMI-1	1	Send poll information by SMS to specified members	Ctm
SMI-2	1	Receive ACK about SMS which is sent	Ctm

SMI-3	1	Extract answer from the received SMS	Ctm
SMI-4	1	Get the responses by SMS and Save to database	Ctm
SMI-5	1	Log sent and received SMS information	Sys
SMI-6	1	Do not store duplicate votes and inform the user	Ctm
		Email Module Interaction	
EMI-1	1	Send poll information to specified members by email	Ctm
EMI-2	1	Provide security	Ctm
EMI-3	1	Receive emailed answers and Update database	Ctm
EMI-4	1	Do not store duplicate votes and inform the user	Ctm
EMI-5	1	Check validity of received answer before adding to system	Ctm
EMI-6	1	Encrypt email data and poll's information	Ctm
EMI-7	1	Log send and received information	Sys
EMI-8	1	Log sent Pdf file of polls information	Sys
		Database	
DB-1	1	Avoid redundancy	Sys
DB-2	1	Use proper stored procedures for security	Sys
DB-3	1	Be accessible just from Data Access Layer	Sys
		Security Aspects	
SA-1	2	A login and reception protocol should be proposed	Ctm
SA-2	1	Application, its modules and services support security	Sys

5. Architectural Design

Guiding factors in defining architecture

Careful analysis of requirements leads to derive following guiding principles that were considered during system architecture finalization:

- There is a business need to exchange data from one application to another (for example, from web-DPS application to SMS Gateway application for SMS sending purposes) in real-time.
- There is a need for message transformation (change of data format or combining of multiple messages into a single message or so) and routing.
- There is a need for the systems to be loosely coupled through asynchronous messaging.
- Re-usability of the interface, so that same data needs to be sent from one system to multiple systems.
- Process integration (Process orchestration) where in a business transaction spans over multiple systems and message feeds need to be processed from one system to another.
- Need of guaranteed message delivery.
- Need of security in application interactions for business purposes.
- Smooth and effective error handling mechanism.

How the DPS architecture originates from above guidelines

During designing the architecture we have considered DPS as a software system which consists of three software applications

- SMS Gateway
- web-DPS application and
- Email server.

For scalability, loose-coupling and technology independence we have introduced middleware-the core of our system that takes care of message transformation, routing, hides the complexity and interaction among the above depicted three software applications.

Middleware provides hub and spoke architecture (Figure 5-1: **Hub-Spoke Model and System Architecture**) for the system where middleware is at the hub and software applications are at the perimeter of bicycle-wheel linked with hub through spoke called 'adapters' in software terminology.

Software applications do not need to bother how the messages should be exchanged among themselves. Rather it's the middleware who takes care of low level technical details and

communicates with each application using different technologies.
As middleware communicates in different manner with different applications, applications are built in any platform and technology.

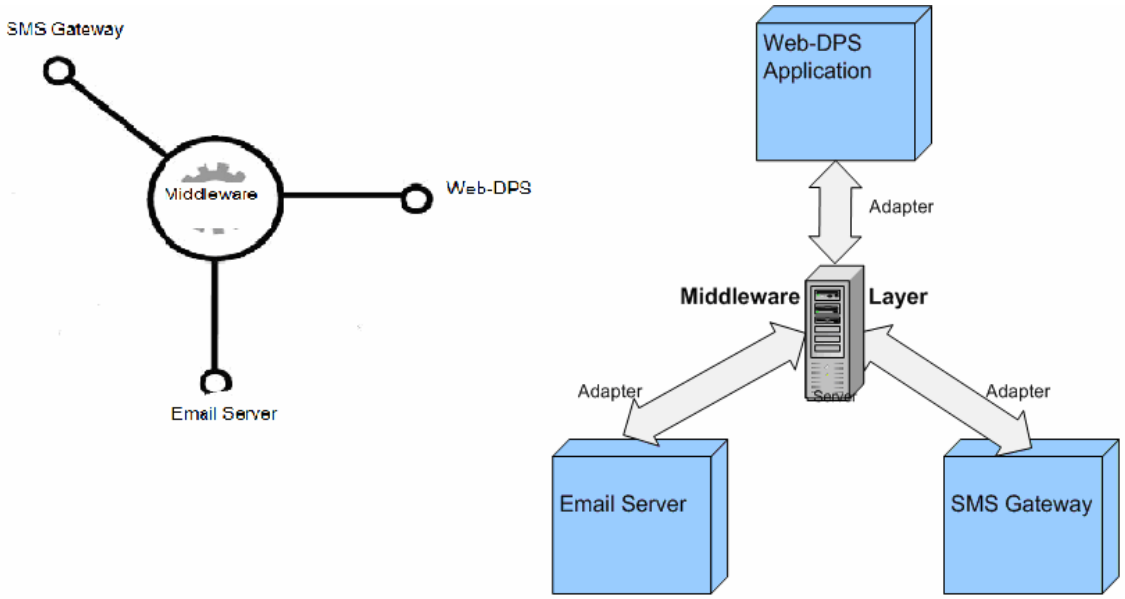


Figure 5-1: **Hub-Spoke Model and System Architecture**

Future considerations:

As per the requirement, in future, if any other mode of communication with the members to be incorporated (for example voice mail), we need to add one more application to the middleware as depicted in above diagram (*Figure 5-1: Hub-Spoke Model and System Architecture*). No need to change whole system. It's only a plug-in to the existing system.

At high level, DPS requires three main business processes as follows:

1. Intimate Poll Details
2. Capture SMS Response
3. Capture Email Response

Intimate poll details intimate the members through SMS and email. *Capture SMS Response* and *Capture Email Response* collect member's voting responses for business decisions and updates at web-DPS application for vote calculation.

5.1 System level Interface

According to **Figure 5-1: Hub-Spoke Model and System Architecture**, software applications are connected through spoke with the middleware at hub, instead of point-to-point connection among themselves.
Each interaction between an application and the middleware specifies an interface in DPS.
Figure 5-2: **System Level Interface** delineates each step at system level as sequence of activities to fulfill the above specified business processes. Step-1 to Step-4 fulfills ***intimate poll details*** business process. Step-5 and Step-6 are for ***capture SMS response*** and ***capture Email response*** business process.
Interestingly, it also delineates single and parallel steps (in different legend) applicable for the business processes. Single steps are sequential in nature and parallel steps can occur in parallel based upon message triggering from initiating software application.

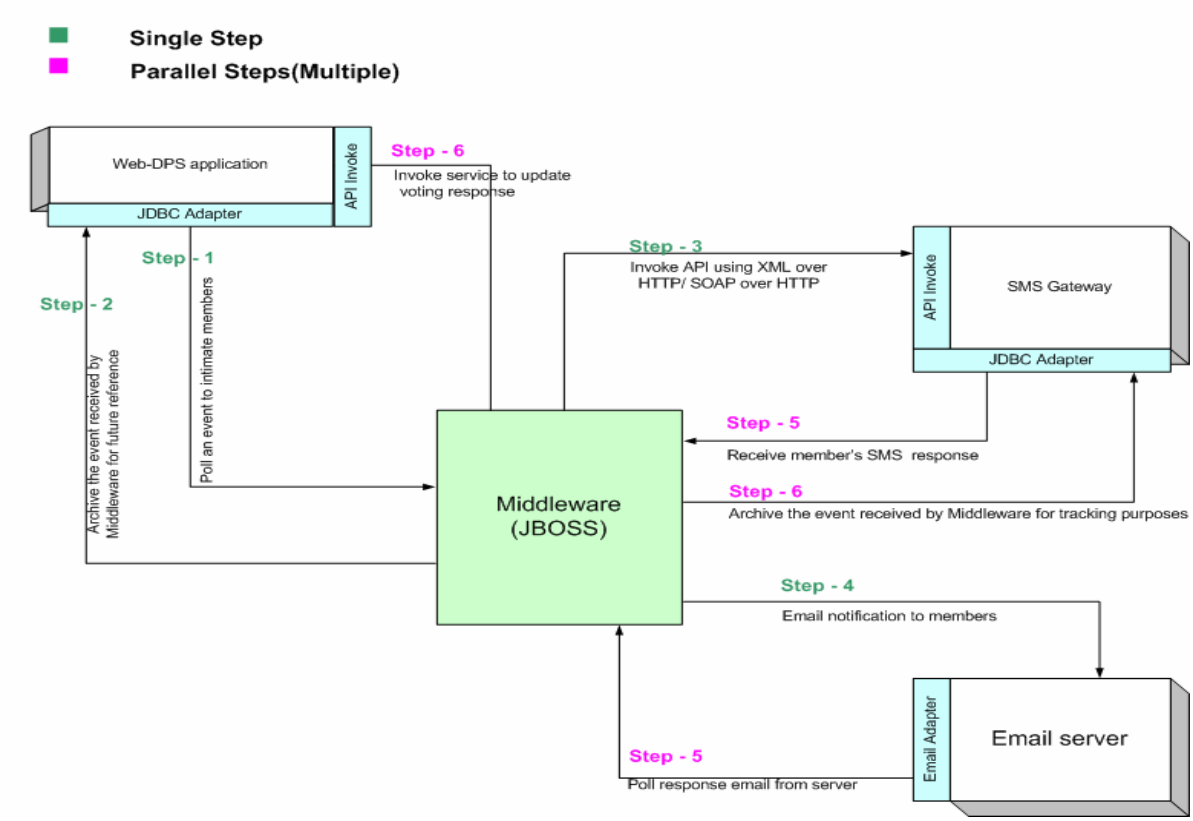


Figure 5-2: **System Level Interface**

5.2 Business Process level Interfaces

At specified above, DPS requires three main business processes as ***intimate poll details, capture SMS response and capture Email response.***

We will specify here sequence diagrams that delineate business process level interfacing requirements.

Intimate Poll Details

As depicted in the below UML sequence diagram, Poll created through web-DPS application will be picked up by listener thread and will be intimated to members through SMS and email.

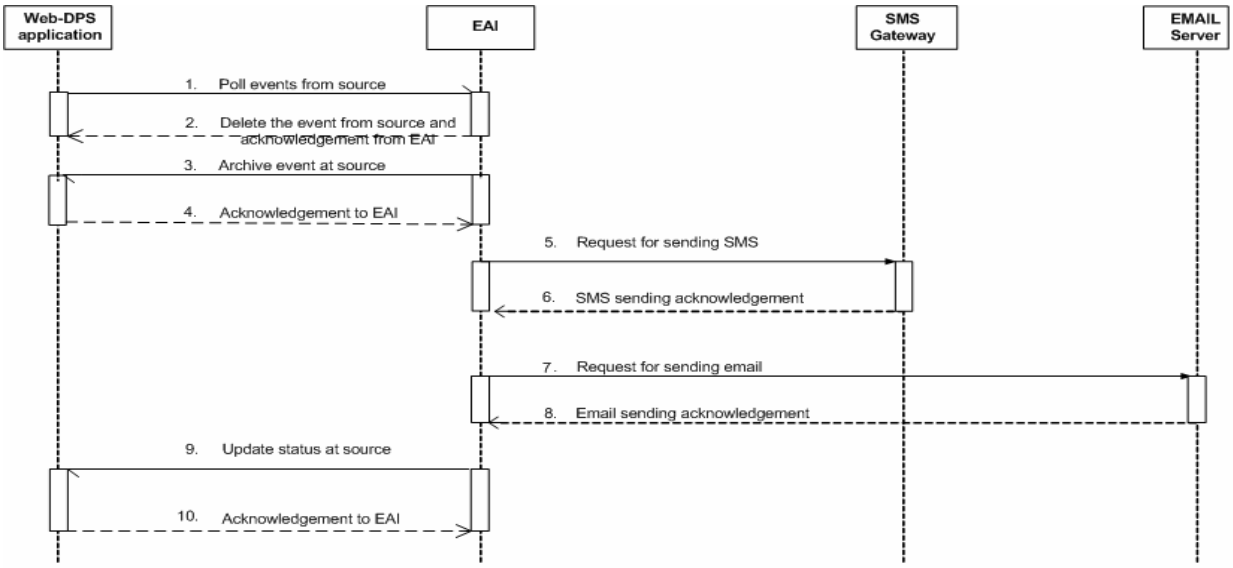


Figure 5-3: **Intimate Poll Details**

Similarly, interfaces for *capture SMS Response* and *Capture Email Response* business processes have been constructed to fulfill system functionalities. (due to page limitation in SCORE report, we are not depicting here)

5.3 Layered Architecture stack of Web-DPS Application

Web-DPS application is a software application in DPS and layered architecture of the same is delineated below (Figure 5-4: **Layered Architecture Stack of web-DPS Application**).

This custom built application provides members to compose business issues. Also it performs major functionalities like maintaining business data in repository, collecting voting responses, voting calculation and so on.

The application consists of *GUI layer, Control layer, business logic and integration layer, data access layer* and *database layer.*

GUI layer enables members to create business issues and to submit, along with much other functionality.

Control layer captures data from web-page and invokes appropriate business logic layer.

Business logic and integration layer implements validations, business rules and invokes data access layer for permanent storage requirement within the application.

Data access layer builds few API's that are the only path to access database from business logic layer and encapsulates the database layer from improper use.

Finally it is the database layer that stores and maintains business data in its repository.

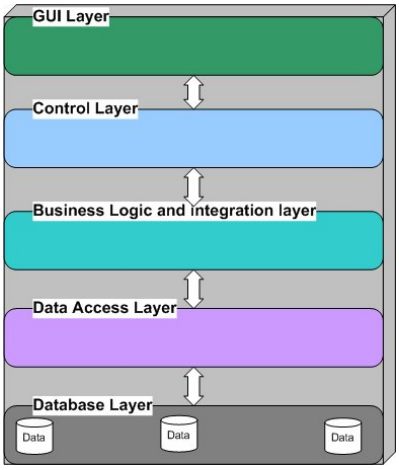


Figure 5-4: **Layered Architecture Stack of web-DPS Application**

5.4 Application Connectivity Choices

There are different technological means by which software applications (web-DPS application, SMS Gateway and Email server) are connected with the middleware at hub (in hub and spoke architecture).

The following section analyzes different technological aspects that are used to connect to a particular software application and makes a comparative analysis to finalize the best suitable choice for each application in DPS.

Middleware layer here is represented as data integration layer or enterprise application integration (EAI) layer.

5.4.1 Web-DPS Application

Four different approaches were considered for integration of web-DPS application with EAI layer.

1. **Integration using XML/HTTP or HTTPS**
   - Technologies used in 'business logic and integration layer' of web-DPS application allow invoking an exposed service at EAI layer using XML message over HTTP/HTTPS protocol. (web-DPS Application→EAI Layer)
   - Similarly, from EAI layer also, exposed service at web-DPS application is invoked using XML message over HTTP/HTTPS protocol (EAI Layer→ web-DPS application)
2. **Integration using Web-Service**
   - An EAI layer service is exposed as web-service and is invoked from 'Business logic and

integration layer' of web-DPS application as web-service (web-DPS Application→EAI Layer)

- Web-services is implemented at 'business logic and integration layer' of web-DPS application as business services or work-flow and the same is consumed as WSDL document through HTTP transport (EAI Layer→ web-DPS Application)

3. JMS Approach

The web-DPS application triggers and receives business messages over JMS queues on J2EE compliant application server.

JMS listener plug-in of JBoss middleware ESB allows listening over JMS queues and triggering middleware ESB services using those messages.

4. JDBC Adapter approach

JDBC listeners can be used to poll a SQL event table at web-DPS application and the listener thread can trigger JBoss ESB service with the same event.

Comparative study of Connectivity Choices

Following table makes a comparative study among all the above discussed connectivity approaches.

Table 5-1: Compare connectivity choices

Feature	XML/HTTP or HTTPS	Web-Service	JMS	JDBC Adapter
Synchronous communication	Yes	Yes	No	No
Asynchronous communication	No	No	Yes	Yes
Supports guaranteed delivery	No	No	Yes	Yes
Type of coupling between DPS application and EAI layer	Loose	Loose	Loose	Loose
Development Time	High	High	High	Less

Approach we have chosen for web-DPS application interaction

Web-DPS application Outbound:

By considering all aspects, we have finalized JDBC adapter approach for web-DPS application outbound communication where web-DPS application will trigger an event to a custom SQL event table located in web-DPS application database.

An EAI listener thread will poll this table and will pick up an event as soon as it is inserted in this table. The event is then dispatched to a subscribed service at EAI layer.

Web-DPS Application Inbound:

A custom built PL/SQL service at web-DPS application end is invoked synchronously from EAI layer. EAI layer receives a response from web-DPS application after updating voting response received from members.

5.4.2 SMS Gateway

- During sending poll details to members, project requirement suggests receiving an acknowledgement from SMS gateway to track whether a member is reachable in his mobile or not.

  To fulfill the same, we have decided to invoke the service built at SMS gateway end, synchronously from EAI layer to receive acknowledgement in the same thread. (EAI Layer→SMS Gateway)

- Project requirement suggests receiving SMS responses. But there is no synchronous requirement.

  SMS gateway can collect SMS responses from members and put in a custom SQL event table configured in its database. We use JDBC adapter approach where a listener thread can poll the event table, pick up events from that table and send to EAI layer for further processing. (SMS Gateway→EAI Layer)

Approach we have chosen for SMS Gateway interaction

SMS Gateway Outbound:

JDBC Adapter approach

SMS Gateway Inbound:
Invoke SMS gateway PL/SQL service synchronously

5.4.3 Email Server

Project requires to intimate poll details to members through email and to collect the responses.

- Sending email to members will be handled using services built at EAI layer to communicate with Email server (EAI Layer →Email server)
- Receiving email from email server will be handled using Email adapter that can poll an inbox of an email account (for example, response@student.mdh.se) (Email server→ EAI Layer)

Approach we have chosen for Email Server interaction
Email Outbound:
Using custom built service at EAI layer.
Email Inbound:
Email adapter to poll inbox of response email account

6. Implementation

DPS is a complex system and its robust, scalable architecture has led team members to build a number of reusable components. It was a challenging task for DPS team to integrate all these components and successfully test the whole system.

For SMS sending we were supposed to use telecom service provider's (like Vodafone, Telia, AT&T and so on) SMS Gateway that takes care of sending SMS to its mobile subscribers through SMSC. For email sending we were supposed to use our MDH university email server.

We faced hurdles while looking for local telecom service provider for its SMS Gateway to use for our project purposes. We contacted around three local service providers for the same and it did not produce any result. Later, we found out that there is one more option in which we can use our *mobile handset* as *SMSC* that will handle sending SMS to any mobile handset roaming around the world and our local SMS Gateway software can communicate with the SMSC. Each mobile connection has its base location (the area where the SIM is purchased) and a service provider (the telecom operator who is servicing that customer). So when a subscriber goes out of his base location (to anywhere in the world), it's a roaming service for that subscriber. In our prototype implementation we used our Sony-Ericsson z550i mobile handset as SMSC application and connected that with our java based software through COM port at SMS Gateway.

We also faced issues while looking for university email server. We were in need of SMTP email server IP, log-in credential details and access to port 25. Unfortunately, due to having strict security rules, university information technology department did not allow the same for our project purposes. Finally, we managed to download, install and configure Microsoft WINMAIL server v4.6.

During designing the architecture although we introduced middleware for scalability, we did not have much expertise in that domain. It was again a challenge for us to understand and implement message transformation and routing in real time. While choosing for middleware, we found that a lot of software like IBM websphere, webmethods, tibco and so on are available. However, all these software need to be purchased from the vendor. So we started looking for open-source middleware and we found JBOSS middleware which suited our requirements.

For web-DPS application designing and implementation, we had a few team members with some experience in JSP and struts. Database requirement was fulfilled by open-source MySql 5.0. We designed a layered architecture for this web-DPS application looking from scalability and security perspective. In data access layer, we built few API's which were the only API's to access database from business layer. As it was a business critical application and requires security, HTTPS provides secured access to the webpage and log-in details are authenticated at server side, transferred as encrypted data to the server.

Now we will depict here few component implementation details for "Distributed Polling System" and how it provides technology independence and loose coupling.

- P**oll Creation:** We have used JSP and struts for web application implementation. We have created an event polling SQL table in the application database from which the middleware is continuously polling for new events. Once a business topic is created in the web, it is populated in this SQL table and middleware interacts with the SMSC application and email server for notification/intimation as per the business rules.
  Note: Technology used for implementing the web based application has no tight coupling with the middleware and instead of JSP and struts we could use any other technology here. The only connectivity between web based application and the middleware, is the SQL table which can reside in any relational databases.

- **SMS Intimation:** We have a JAVA service that connects COM port to Sony-Ericsson z550i mobile handset (acting as SMSC). From middleware we are invoking this java service synchronously to understand whether SMSC has successfully sent SMS or not.
  Note: We have implemented here one java service/API to communicate with the mobile handset acting as SMSC. But it could be any service implemented in any language like C, C++ and so on.

- **Email Intimation:** Similarly, we have another java service that sends email to the specified list of email-addresses. From middleware we are invoking this service to send email notifications. Here some business rules decide whether email will be sent as plain-text message or as PDF attachment.
  Note: We have implemented here a java service to communicate with email server. But it could be any service implemented in any language like C, C++ and so on.

- **Response Update:** For all voting responses captured through SMS and email, we update back to web-DPS application by invoking one PL/SQL API. This API maintains all business rules, duplicate verification and so on.
  SMS: SMS responses captured through SMSC are put in a native SQL database table from which middleware polls the events and updates in the web-DPS application by invoking the PL/SQL API.
  Email: JBOSS Email listener polls a specific inbox of an account and updates in the web-DPS application by invoking the PL/SQL API.

- **PL/SQL API to update voting response in the application**
  A service/API has been built that implements duplicate voting verification, verifies SMS response from authenticated user, proper type of voting mechanism. This API is built in PL/SQL and is invoked from middleware layer synchronously to update voting response in the application.
  This service/API could be built at business layer also using different technologies like java and could be invoked from middleware layer to update the response in the similar manner. It again emphasizes scalability, technology independence and loose coupling.

- **Daemon process for poll result announcement at deadline**
  There is a daemon process that monitors the deadlines for all active polls and as soon as a particular poll reaches its deadline, the daemon process triggers another process that calculates and announces the result.
- **Process for poll result announcement before deadline**
  There is also a process that verifies whether a particular option has achieved more than 50% of votes or whether all members have voted before deadline or not. If any such kind of rules is met, this process calculates and announces the result.
  This process is triggered on voting response update in the application from external channel (i.e. from SMS or from email).

Following section describes the algorithm for calculating votes, received from members through SMS and email.

Algorithm for vote calculation

On the basis of options available to users for voting, the polls have been classified as:

a) Simple Multiple Choice
   i. Single selection (Yes/No or Multiple options): Radio Button / Combo box
   ii. Multiple selection: Check Box / Drop Down List

b) Priority based / Rank order Question
   i. Order wise
   ii. Weight Base

c) Matrix Based (Multiple questions with combination of above mentioned schemes)

On the basis of result announcement:

a) Declarative Polls: In Declarative polls there is no single winner, usually created for trend or survey purpose to collect opinion about some subject

Ex: What is your opinion on the canteen facilities?

A) Excellent B) Good C) Needs Improvement D) Need Replacement

Result Announcement: 10 % voted A) Excellent || 30% voted B) Good || …

b) Decisive Polls: In Decisive polling a single clear winner is announced

Ex: Result Announcement: Mr. K Narayanan is winner for polling of CTO (Chief technology officer) position.

Considerations for algorithm efficiency:

The result calculation process does not need to start unless minimum votes required to produce a winner are voted/casted (yes/no option do not start unless 50% of the overall voters have casted their votes).

Use of cache mechanism to store voting results instead of accessing database for result each time a vote is casted.

Announce the result as soon as we got a clear winner. Don't wait for all 100% voters to vote.

Algorithm

t_{start} : Start date & time for poll to open for voting

t_{end}: End date & time for poll to close

t_{vote} :Date & time when the user vote is received.

If the user has voted:

a) Before the voting starts($t_{vote} < t_{start}$): User will be alerted through the use of same electronic medium.

b) During the voting is open ($t_{start} <= t_{vote} <= t_{end}$).

c) After the poll closes($t_{vote} > t_{end}$): User vote will be neglected.

Algorithm for Result Calculation

a) *Yes/No option*: If any option has got more than 50 % of votes winner is announced and poll is closed (even when all users have not voted).

b) *Single Choice/Multiple Choice*: Result cannot be predicted unless all users have voted or the poll end time is reached. The vote getting maximum number of options is the winner.

c) *Simple Priority/Rank based:* It is a declarative type of poll, in which the percentage or number of voters trend is announced when either all users have voted or poll end time is reached.

d) *Weighted Priority Based*:

V_p (Value for i th vote option) = $\sum W_i$ ($1 <= i <= n$)

= Sum of all weights given by users to this option

The vote option that has maximum Value V_i is the winner.

e) *Matrix Based*: Combination of above mentioned logic for matrix based multi question polls.

In the below section we discuss about the security aspects for DPS from software application and infrastructure set-up point of view.

Security Aspects

There are different ways to set-up servers for our applications:

- Assume the servers for web-DPS application, middleware, SMSC and email server are situated in the same LAN protected by firewall.
  Since the webpage can be accessed over internet anywhere in the world, to provide secured access, we can implement HTTPS to access the webpage and log-in details, other credentials are authenticated at server side, transferred as encrypted data to the server.
  As all the servers are in same LAN secured by firewall, so interaction among the applications through middleware software server does not require security.
  In telecom network, IMSI and MSISDN are the two unique parameters across universe. Our application maintains these two values against each member and while receiving SMS response from telecom network, we verify against these two parameters for authenticating SMS vote.
  Email voting response received from members should be verified using digital signature at email server level to assure authenticate member's vote.

7. Verification and Validation

7.1 Requirements verification and validation

Requirements testing ensure that the requirements have been interpreted correctly, are reasonable, and are recorded properly. In software industry different approaches are being used for requirement verification and validation like: prototyping, tracing approaches, user manual writing, reviews and inspections [1, 2]. We used the Review approach for requirement verification and validation. In regards of this approach we read and analyze the requirements from the SCORE document, then conduct internal team meetings, discuss the problems and agreed on actions to address these problems. Meanwhile, we also arranged meetings with internal customer i.e. project supervisors to validate the requirements. As an external customer was not involved on site during development span but he was involved through electronic means of communication. For requirement validation we clarify requirements from the external customer through emails and he gave us feed back with some changes and finally after the changes were made it was verified and validated by internal customer. Finally, we were able to put requirement analysis to next level of development phase.

7.2 Design verification and validation

7.2.1 Testing approach

DPS is a web-based java application. The testing approach adopted in the project is in two ways.

- Functional testing
- Compatibility of the entire application with the technical environment

7.2.2 Functional Testing

The functional testing which is done for requirement check, whether all functionality of SCORE are implemented in the project. Functional testing is based on the requirement specification documents. On the basis of use cases the test specification document is prepared .Application is tested manually. Functional testing is done in three levels.

- Unit testing
- Component testing
- System testing

The detailed information about the functionality that is tested is described in the below table:

Table 7-1: Functional Testing Levels

Sr No	Type of Testing	Functionality Tested	Result
1	Unit	1. Create User 2. Create Poll 3. Send Email 4. Send SMS 5. Reply vote through Email thick client 6. Reply vote through PDF file 7. Announce Results 8. View Poll Result….. and 15 more functional units	Passed
2	Component	1. Email Generator(Send and Receive) 2. PDF Generator 3. SMS Sender and Receiver	Passed
3	System	Testing the application on basis of complete software	Passed

7.2.3 Compatibility Testing

We used the testing approach i.e. testing by web page. The application is deployed on local server. We are using the Tomcat 6.0 as a web server and JBOSS application server environment and database is also deployed on same server under Windows environment. We performed the compatibility test for our application to tests whether it is capable of running on different browsers or not. We tested our application on Internet Explorer, Mozilla FireFox and Google chrome and it runs very smoothly.

8. Development Process

We have used agile methodology and in particular Scrum for development of our project.

Scrum requires frequent and sometimes face to face interaction with customer which was not possible for us. Thus we assigned the role of Customer manager (who will act as customer for our project) to one of our members. One person is selected as the Scrum Master. We reserved first four weeks for Planning, Analysis and Design. During these four weeks we came up with an initial version of task/story list, core architecture and partial design of the system. Initial task list consisted of up to 54 different tasks that had grown to 132, when we finished project implementation. Each task was assigned a priority (on scale of 1 to 5 from urgent to optional respectively), task description, remark and an example scenario.
The project was developed in multiple sprints; each lasted 4 to 5 days of around 20 hours each. Before starting each sprint we had a Sprint meeting and before starting each days work we had a Scrum/standup meeting (max of 10-15 min).

Sprint meetings: In this, our Customer manager selected those stories which had maximum value to the project, which were more risky and some were those on which many other tasks were dependent. Our first sprint consisted up of tasks email and SMS communication.

Standup meetings: In the standup meeting, the Scrum master selects the tasks from the sprint, which are to be finished on the current day. Members or group are assigned tasks from the task list and their names are written along with the tasks on the whiteboard. For some critical tasks we even assigned 2 members (pair programming for example we did it in task: result calculation for polls). The members then start implementing them. On the next day standup meetings all members stand together for short time and each one speak the following three things:

What tasks the member has achieved after the last meeting?
What are the problems they are facing?
What tasks the team members are planned to do now?

Once the standup meeting is over, the scrum master updates the excel sheet of story list.

Then the scrum master discusses the problem faced by individuals. Our experience is that during the first sprints we hesitate to communicate with each other. But in later sprints team members start proactively helping each other.
After finishing intensive sprints we celebrated two parties and after finishing the implementation phase we had a group picnic.

9. Project Plan

9.1 Activity plan

Following activities are planned according to the Sprint. A single sprint consists up of 20 hour work per person for 6 members. Our assumption is each sprint will last for one week.

Table 9-1: Activity plan

Activity	S1/w 45	S2/ w46	S3/ w47	S4/ w48	S5/ w49	S6/ w50	S7/ w51	S8/ w52	S9/ w53	S10/ w54
Project preparations & planning										
Requirements analysis & definition										
Architecture and design										
Environment Setup										
Communication layer implementation										
Business layer implementation										
GUI & database layer implementation										
Unit testing & Integration testing										
Functional testing										
System and Alpha testing										
Project Report										

S1: Sprint 1, W45: Week 45th
Major activity: Dedicated team members assigned for the task and will be focus for the sprint
Minor activity: This can be done along with major activity and not the main focus of the sprint
Optional activity: They are not predictable at time of planning but might happen during the sprint.

Implementation will not start till the 4th week. As their will be continuous integration of modules unit testing and integration testing will go hand on hand with implementation. The acceptance (alpha) testing can only be done once the system is stable.

Project group
Following are the main responsibility of the team members. But we work as a team and each member will be involved in every phase of the project.

Table 9-2: Group members and their responsibilities

Name	Initials	Responsibility (roles)
Jenis Kavadiya	JK	Project Manager, Scrum Master
Avijit Dutta	AD	Architect, Developer
Aparna Vijaya	AV	Database Designer, Documentation Coordinator
Farahnaz Yekeh	FY	Test coordinator ,Developer
Rishabh Gupta	RG	SVN Coordinator, Developer
Sajjad Ali Khan	SK	Developer, Designer

The following table lists the major risks we have projected, their severity and the way we overcome them.

Table 9-3: Project Risks

Risk	Impact	Risk Mitigation and prevention
Time shortage	High	Use of Agile (scrum) process, Proper planning.
Competence in technologies	Medium	Knowledge sharing sessions and Training. Pairing less competent members with more experienced one.
Miscommunication	Low	All main communications and decisions will be written down in MOM's and circulated through and Emails and Google group.
Server crash (database/SVN)	Low	Generation and updating the database scripts periodically. Regular backups.

10. Management Plan

The DPS process involves different activities. These activities need to be distributed and managed among team members. As we are a distributed team, communication among different team members becomes one of the major areas of focus from project management perspective. Various methods are used for proper communication among team members. Weekly formal meetings using video conferencing was organized to discuss weekly status report of tasks assigned to individuals. Google groups, Skype chat are other tools we used for communicating quick issues.

We have various scrum meetings, start-up meetings during implementation phase. For document and code sharing, we used a configuration management tool SVN. One of the team members is assigned as SVN coordinator whose responsibility includes managing directory structure of the SVN repository, taking regular backups to counter server crashes. The access to the repository is strictly restricted and team has specific access rules defined for them. All these rules and access information is maintained in a SVN policy document. We use Google groups for creating topics regarding ongoing activities and for discussing and providing feedback about other's work in the team. In addition to all these, we have different meetings on requirement basis such as knowledge sharing sessions.

All formal meetings are documented by writing MOMs. Each MOM has clear description of the meeting and actions assigned on individuals before the next meeting. We also have weekly meetings with DSD course supervisor which includes review of tasks done in previous week. We also had two project status presentations as part of the DSD course. This presentation includes discussion about all challenges we faced, activities we have completed, and planning for the coming weeks. Using all these methods, we kept track of all activities and ensured that we are working on the correct track, kept every team member on same level and pace. We have created different roles for different activities. Every team member has ownership of task corresponding to the role he has assigned to. For example one of the team members was assigned as document manager and all project related documents are his responsibility and he is the one who keep track of status of the documentation tasks assigned to other team members. This way we have created sense of ownership among team members which really helped us a lot during the whole development process.

We used a structured approach for communication among the team members. All members were able to contact project manager directly. And only project manager is responsible for communicating with DSD and SCORE supervisor. Any issue and doubt of the team members has to go through project manager.

11. Outcomes and Lessons learned

According to the SCORE requirements we have implemented almost all the functionality of the system and developed a functional prototype. This working prototype is a result of all project activities we have performed in last ten weeks for the SCORE competition as part of Distributed System Development (DSD) course and artifacts including project description, requirement, architecture and design, presentation demos and weekly report are uploaded at http://www.fer.hr/rasip/dsd/projects/distributed_decision/documents

Following is the snap-shot of some implemented web-DPS application GUI's, email receipt, SMS receipt and poll result.

"**Create poll GUI**" – Enables members to create a new business issue.

"**New mail pop-up**"— pops up when a new email received by a member.

"**E-mail as PDF attachment**" – sends business data as content of a password-protected PDF file.

"**Poll receipt SMS**" – Displays the SMS received with content of business data.

"**Poll Result GUI**" – Displays the voting results.

"**PDF content-once open through password-protection**" – Displays business data along with voting options , from password protected PDF document.

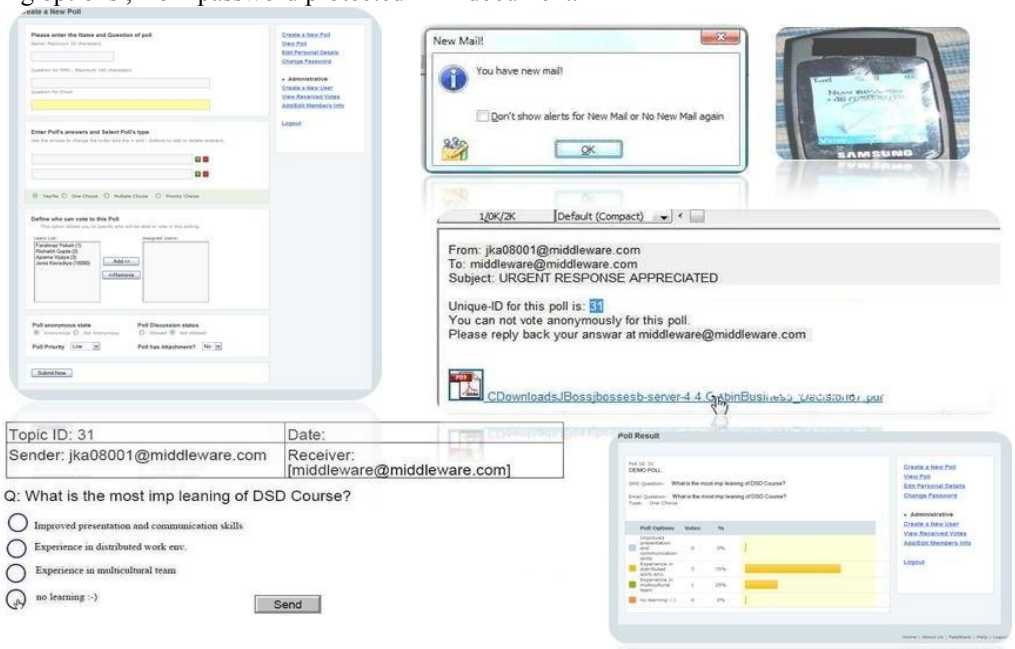


Figure 11-1: **Snap-shot of some DPS functionalities** (create poll, receive mail alert, receive sms, received mail, attached pdf info and view poll result)

The application we have developed has some limitations because of the time constraint we had. It does not support some features such as discussion among poll members and some security features (Digital signature). However, these features were considered during design phase and incorporated in system design and can be implemented as future improvements in the application.

In addition to these we faced many challenges during development process. Main challenge we faced was during requirement engineering. The only way of communication between team and customers is thorough email which led to some misunderstood requirements. The initial requirement definition was created using functional requirement document provided by SCORE. We had misunderstanding in one of the requirement of having anonymous user in the system; this was sorted out late in the development cycle which cost us significant amount of time because the solution of this problem included changing the design (Database and GUI changes). There are many other challenging issues during requirement engineering which took a lot of effort to reach specific design decisions. Initially we had different views for different requirements among team itself. We had various team meetings and mail communication with SCORE supervisor in order to achieve clear requirements with respect to all given conditions and constraints. GUIs verification was another important challenge. Main difficulty we encountered was GUI creation because it

included usage of all possible combinations for poll creation (which was complex in nature. Refer scenarios matrix for details) with simplicity in order to make GUIs user friendly.

We found some problems within the team during early stage of the project because of different communication and cultural difference barriers. Due to time constraints we had some team meeting on weekend which created problem for some team members because they were not comfortable working on weekends because of their work culture. But with mutual cooperation, understanding, respect and helping each other, we successfully overcome all problems. Inefficient competency and configuration problem for different software like email server, email clients, application server, web server and integration of these software with rest of the core application were a major issues in the project. By self learning, knowledge sharing and organizing sessions on various technology related issues we led our development process in a smooth way.
Finally, we are able to finish our job in a smooth and systematic manner.
During the project we have a good chance to learn some new technologies. Meanwhile, we also faced some problems during the system integration due to strict deadlines. Moreover, there was less time for testing of project. By all team working together under one roof, we were able to implement most of the system functionalities in a very limited development time.
The team is able to implement all functionality of SCORE and able to develop a fine working prototype. The project is under DSD curriculum and team of different diverse culture and nationality members are working in distributed fashion to face overcome different real life challenges and work in distribute manner towards one goal. This project can be further enhanced by adding some new pretty good features if there is some more time for the implementation. However, with the current features and functionalities it is ready for the end users to use it in the real environment.

12. Summary

Our project “Distributed Polling System” is a huge learning experience for us that has taught us to invest more time in proper architecture designing which in-turn reduces heavily the implementation time for any system.
In addition to that, the architecture has guided us to segregate the whole system into granular components that has enriched maximum reusability and utmost team activity. Even the architecture has helped us to invest less time for component integration which has become evident while we started getting expected system behavior after our first time component assembling.
In spite of having six team members from different cultural background and practices, by investing twenty hours per person per week for almost ten weeks, we successfully designed, developed, integrated and tested the functionalities.
It’s our honor and privilege to be part of such kind of real time project implementation and we believe we could do further enhancement to this software in different aspects to make it a commercial software.
Although we could not invest much more time due to having parallel courses in the same curriculum, we would be grateful to work further on this software to make it commercially successful.

13. References

[1] William E. Lewis, “Software testing and continuous quality improvement”, Auer Bach Publications-2005
[2] David A. Cook, “Validation and Verification of Requirements”, accessed on 11th January, 2009 from http://www.sstc-online.org/Proceedings/2002/SpkrPDFS/ThrTracs/p961.pdf
[3] “How to Send SMS Messages from a Computer / PC?”, http://www.developershome.com/sms/howToSendSMSFromPC.asp
[4] Linda Rising and Norman S. Janoff, “The Scrum Software Development Process for Small Teams”, http://members.cox.net/risingl1/Articles/IEEEScrum.pdf
[5] Len DiMaggio, “Adapters for an ESB”, http://www.redhatmagazine.com/2008/05/22/adapters-for-an-esb/
[6] “Winmail Mail Server 4.6”, http://www.vista-files.org/programs/amax-information-technologies-inc/winmail-mail-server.html
[7] Burr Sutter, “JBossESBQuickStart”, http://www.jboss.org/community/docs/DOC-10363
[8] “SQL Statement Syntax”, http://dev.mysql.com/doc/refman/5.0/en/sql-syntax.html#nolinkhere