Design Document

E-Elections Software

"The Pericles Project"

Produced by: Team Dragon

Amir Reda Atalla
Mathew Evans
Mike Levy
Cory McKay
Sitai Sun

TABLE OF CONTENTS

I. INTRODUCTION

1.1 Purpose of this Document

This document is meant to outline the design of the Pericles Elections Software. It will be used to give clear guidance to those who implement the software and to ensure that they conform to the overall goals of the Pericles Project. This document will also be used by those maintaining the software in order to help them assess the structural ramifications of any changes that they wish to make to the software. Finally, this document will be used as a means of verifying the design with the client in order to ensure that it meets her needs. This document will include justifications for the design choices that were made, in order to ensure that those who are maintaining the software are aware of why the choices were made.

1.2 Scope of this Document

This document is not intended to be a comprehensive guide to the requirements and goals of the Pericles Project. Although this information is touched upon briefly on in this document, the main focus is on design. The reader is referred to the Requirements Document for any clarification that is needed relating to requirements.

1.3 Structure of this Document

This document begins with a very high level overview of the design, and then gradually focuses in on actual class structures, their interactions and their inheritance patterns. This is then followed with a justification of the design, an exploration of the design's extensibility, an explanation of the testing that will be conducted following implementation and an outline of those extra features which have been incorporated into the design and those which have not been but might eventually be useful. There is also a glossary and a list of references included. The reader is referred to the glossary if he or she is unsure of the precise meaning of any terms that are encountered.

It is not necessary to read the appendices of this document to understand the design of the Pericles Project. They are included, however, because they will be an excellent resource for the implementers and maintainers of the software. They are also a useful resource for readers who desire more precise information on a specific aspect of the project.

The first appendix is an outline of how the GUIs of the various components of the Pericles software will operate. This section will be useful to readers who would like a more in-depth look at user processes than is provided in the use cases of Chapter III. The second appendix gives an outline of the database schema. The remaining appendices are outlines of various format conventions. They are included in this document in order to prevent confusion between different members of the implementation team who are working on different parts of the project.

II. SCOPE

2. 1 System Objectives

The goal of the Pericles Project is to generate a piece of software that will allow elections to be held electronically over a distributed network. Voters will be able to use a generic Pericles Voting Client that will allow them to log onto and vote on elections running on a Pericles Elections Server. The parties holding electronic elections will be able to customize the settings of each election as well as custom design ballots. All data communicated to or from the voters will be secure, so that nobody will be able to vote unless they are authorized to do so and nobody will be able to tell how anybody else voted. The storage of the voting information will similarly be secure and private.

This version of the software is meant to be an initial exploration of electronic voting software. The main objective of the software is therefore to provide a base upon which more sophisticated election software can be built, which is why there is strong emphasis on scalability in the design. The initial release of the Pericles software is not expected to be perfect or fully comprehensive, nor will it be able to handle system loads such as would be encountered in a federal election. It is, however, designed to be easily modified and expanded as the technology, legal issues and ethical matters relating to electronic elections are further clarified.

2.2 Design Constraints

The following design constraints were imposed by the client:

a) All database interactions must be made using Postgres or MySQL.
b) All software must be written in Java.
c) The Voting Client must run in both Linux and Microsoft Windows environments.
d) The Elections Server must run under Linux.
e) Ballots must be described using XML.
f) All configuration files must be in XML.
g) Statistics reports must be in HTML.

2.3 Design Limitations

The following design limitations were decided upon by the design team:

a) All database interactions must be made using MySQL. This decision was made because the implementation team has a much greater familiarity with MySQL than with Postgres, and Postgres has no obvious advantages for this system.
b) All communications between the client and the server will be implemented using RMI. It was chosen out of the three options of CORBA, sockets and RMI. RMI is the easiest to implement, and the advantage of CORBA being language independent is not relevant to this project, since it is a design constraint that Java must be used to implement all aspects of this project.
c) Voters should have access to a secure e-mail address. This is to increase security. In the situation corresponding to maximum security, the Pericles Elections Server will generate a random username and password for each voter, which will be automatically sent out over e-mail. This will help to ensure that no second parties have access to the voter identification information. However, it is recognized that this does impose limits on accessibility. The software therefore does allow

elections where usernames and passwords are specified by the election authorities and access to e-mails is no longer necessary, but exercising this option is viewed as a serious security risk.

d) The Pericles Election Software only accommodates ASCII characters. This decision was made because ASCII is recognized as an international standard and the implementers will not have the time or the knowledge of foreign alphabets to implement alternative character sets.

e) Voters and elections officials must have some familiarity with English. Although the ballots can be written in any language that uses the ASCII character set, the interface elements of the software (such as login prompts) will be in English. This limitation was decided upon because the initial release of the software will be limited to North America because of legal issues related to encryption technology.

f) The system will only be guaranteed to function properly under conditions that can be simulated in the Reynolds 008 computer lab. The testing team will not have access to computer systems or a network environment that could simulate greater demands on the elections software, nor do they have the means of penetrating the University of Guelph firewall to simulate conditions that the system would face in the real world. The testing team and the implementation team also do not have root access to the Reynolds 008 computers, which imposes further limitations of their ability to verify the robustness of the software, particularly in regards to the database administration

g) All questions asked on the ballot must be multiple choice. This is to simplify the collection of statistics so that they can be easily tallied without any human intervention. Future versions of the software may incorporate text fields for use in situations such as surveys.

III. DESIGN OVERVIEW

3.1 Class Diagram

The following class diagram can be used to give an idea of how an election on a Pericles Elections Server will operate, on a very high level. The remaining sections of Chapter III will further expand the some of the concepts introduced in this diagram. This diagram is not an outline of the actual implementation structure of the software, but is rather an introduction to the higher level concepts relating to elections. Please see Chapter IV for diagrams linked to the actual implementation of the software.

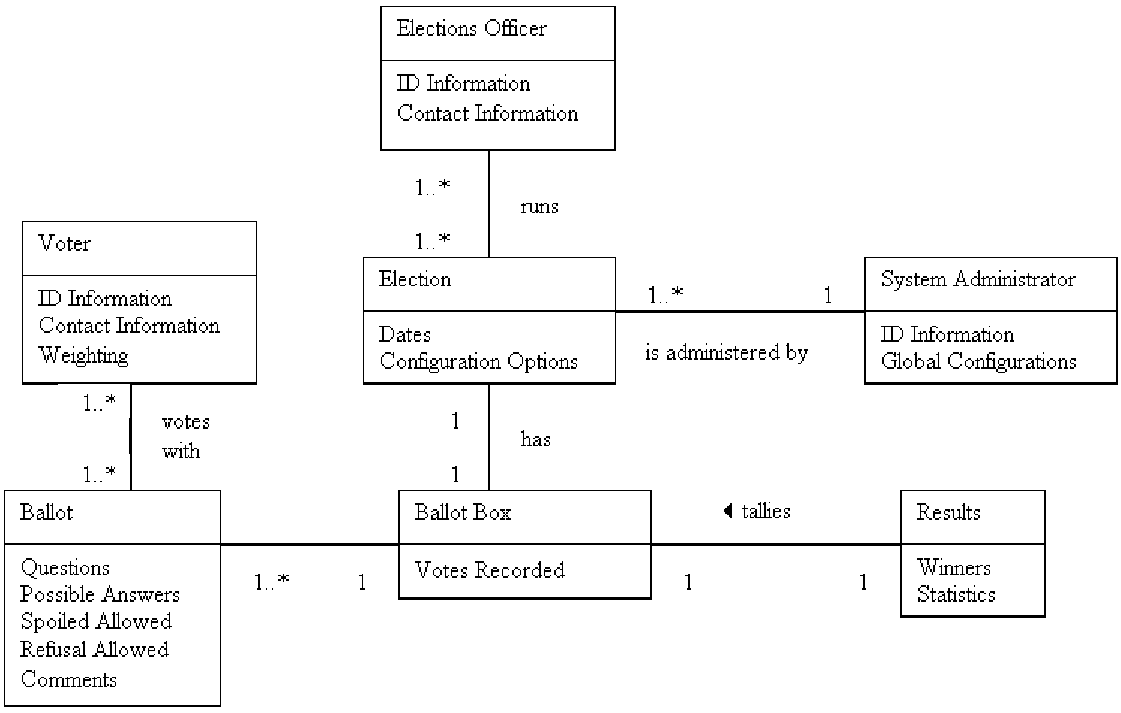

Figure 3.1: High level class diagram of an election.

3.2 Human Users

The following classes of people will be involved in the use of the Pericles Election Software:

3.2.1 System Administrators

A system administrator will be required to oversee the installation and overall operation of each Pericles Elections Server. System administrators will not have control over or access to any particular elections once they are started on a server, but they will be the only ones able to authorize Elections officers to start new elections. They will also be involved, along with elections officers, in suspending and restoring elections.

Systems Administrators will also be able to impose global maximums on how many people can vote in each election and how long election data can be held. This will be done to prevent single elections from overloading the limits of the systems that they are installed on.

Systems administrators will inevitably have root access to the systems that elections are run on, so they constitute an important security risk. As a precaution, data will be stored in such a way so as to prevent them from being able to tell how individuals voted.

3.2.2 Elections Officers

Elections officers are impartial individuals who are given responsibility for overseeing individual elections by the organizations holding each election. There may be one or more elections officer per election, and there will be a customizable minimum of elections officers that will be required to enter their login information to alter any elections settings. Elections officers will be responsible for writing ballots, configuring elections settings, entering enumeration lists of authorized voters, posting election results, suspending and restoring elections in the case of problems that arise and answering questions from voters by e-mail.

3.2.3 Voters

Voters are those people who are authorized to vote electronically by elections officers. They will use the Pericles Voting Client to do so.

3.3 Software Components

The Pericles Election Software will be made up of several software components. See Appendix I for a more detailed description of each of these programs in the context of their graphical user interfaces. The main components are as follows:

3.3.1 Pericles Elections Server

The elections server will be made up of three executable components that interact with the MySQL database which stores all of the election data:

3.3.1.1 Pericles Elections Administrator

This program will be used by the systems administrator of each system to authorize elections officers to create new elections, suspend elections that are in progress when something goes wrong, restore elections that have been suspended, set global limitations on all systems that are in progress and change the systems administrator passwords. The elections officer(s) concerned must also enter their identification information for all but the last of these functions. This is to prevent the systems administrator from having too much individual power over elections. This program will also allow the setting of how many elections officers must log in at a time in order to edit election settings.

3.3.1.2 Pericles Elections Editor

This program will be used by elections officers to create and edit (this can only be done before elections actually start) election settings as well as change their passwords. This includes configuring the details for each election, designing ballots and authorizing voters. The settings for each election will be stored on the database. Under the default setting, this program will also generate information such as voter logins and passwords and store them in the database. The elections editor will also formulate ballots as they are created and save them in XML form on the database. The settings that can be configured are:

a) Date and time to send e-mails: when to send e-mails informing voters of the information that they need to register their votes.
b) Date and time that voting begins: The date and time that voters may begin to have their votes registered.
c) Date and time that voting ends: The last date and time that voters may have their votes registered.
d) Date and time to publish results: The date and time that the results of the election will be released.
e) Date and time to remove voter information: The date and time that all voter information is deleted from the system.
f) Time window: The amount of time leeway that voters will be given to enter their votes because of differences in system times on different computers.
g) Allow voters to change their votes: This option will allow voters to change their votes if they have already voted.
h) Allow weighted votes: This option will allow elections to be held in which different voters have differently weighted votes, such as shareholders' elections.
i) Username specified: This option will allow usernames to be specified in the enumeration list rather than randomly generated, as is the default. This is to accommodate situations such as surveys where voting does not need to be done anonymously or where it is desirable to use existing information such as a valid student numbers as identification information.
j) Password specified: This option will allow passwords to be specified in the enumeration list rather than randomly generated, as is the default. This is to accommodate situations where it is desirable to use passwords that already exist on other systems.
k) Do not require password: This option allows for elections where voters do not need passwords, such as surveys.
l) Do not distribute ID information by e-mail: This will disable the default of sending election codes, user names and passwords out over e-mail to voters. Elections officers will then be responsible for ensuring that voters are made aware of all of the information that they need to register their votes.
m) Allow refusal of ballots: This option allows voters to refuse their ballots before reading them.
n) Allow spoiled ballots: This option gives voters the option of spoiling individual questions on their ballots.

o) E-mail address of elections contact: The e-mail address that will be distributed to voters so that they will have someone to contact with any questions that they might have. This will usually be one of the elections officers' e-mail address.
p) Ballot comments: Comments that will appear at the top of all ballots in an election.
q) Question comments: Any comments preceding any questions.
r) Question: The texts of questions that voters will be asked to vote on. There may be an arbitrary number of questions per ballot.
s) Choices: The options that correspond to each question. There may be an arbitrary number of choices per question.
t) Statistics to be reported: The statistics that will be reported at the end of each election. The options are the text of the winning choice, the total votes for each choice and the percentage share for each choice.
u) Location of statistics: Where to e-mail the web pages containing the results of elections.
v) Authorized voters: The elections officer(s) will enter a file containing the enumeration list of all voters that are authorized to vote in each election and any related information. See Appendix V for more information.

3.3.1.3 Pericles Elections Generator

This program will be responsible for handling the mechanics of all elections. Once it is running, it will check the database regularly to see if any actions need to be taken. These actions include checking to see when e-mails need to be sent out to voters, when votes should start being accepted for a new election, ensuring that the system is working properly, when votes should stop being accepted for a new election, when to send out statistics for an election and when to delete information from the database.

When a new election is ready to be activated, the Elections Generator will create a unique Server Object for that election. This object will be responsible for sending out e-mails to voters and elections officers, validating voters who attempt to vote, sending uncompleted ballots to voters, parsing completed ballots, recording votes on the database and acknowledging votes by generating and sending voters verification numbers. The Server Objects will handle all communications with Voting Clients, including the encryption and unencryption of messages.

The Elections Generator will be responsible for creating, destroying and telling Server Objects when they need to take scheduled actions. The Elections Generator will also check on these objects regularly to make sure that they still exist and are accepting votes. This is to help ensure that there have been no security breeches. If there is a problem, the Elections Generator will send an e-mail informing the elections officer(s).

3.3.2 Pericles Voting Client

This is a generic piece of software that can be used to interact with any election running on a Pericles Elections Server. It will send voter authentication information to Elections Servers, receive ballots, interpret and display the ballots, allow voters to enter information on the ballots, return the ballots to the server and display verification numbers and error messages returned by the server to voters. It will also encrypt and unencrypt all communications with servers.

3.4 Use Cases

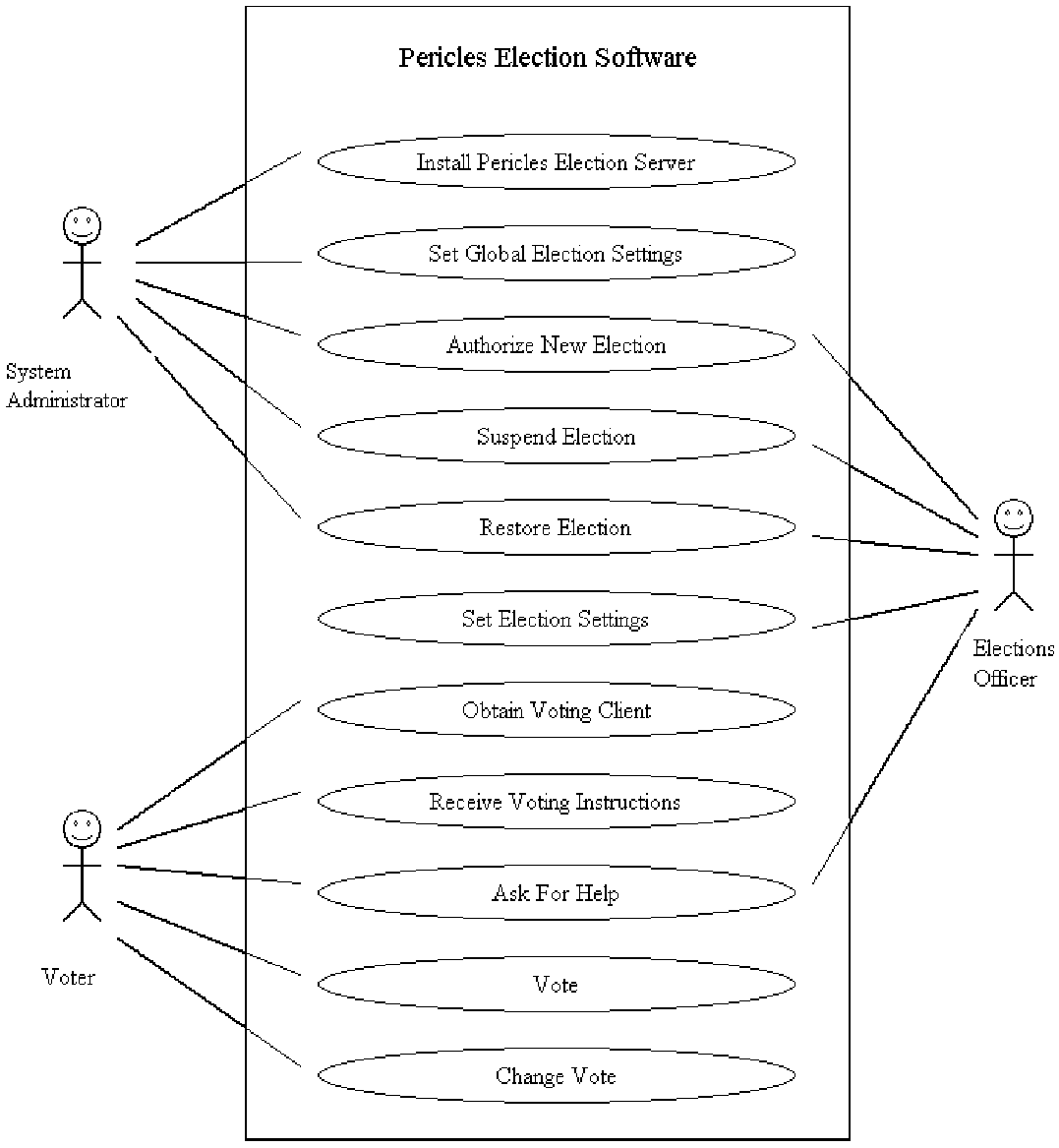

Figure 3.2: Use Case Diagram

Use Case: **Install Pericles Election Server**
Actors: System Administrator

Type: Secondary
Descriptions: The systems administrator acquires the Pericles Election Software and copies it onto a computer that is connected to a distributed network and has a secure MySQL database up and running. The systems administrator uses the installation software provided to set up the Elections Generator and enters the settings on the configuration file (see Appendix VIII). The systems administrator then runs the Elections Generator.

Use Case: **Set Global Elections Settings**
Actors: System Administrator
Type: Secondary
Descriptions: The systems administrator runs and logs in to the Elections Administrator. He or she then defines global rules that apply to all elections on the server.

Use Case: **Authorize New Election**
Actors: System Administrator, Elections Officers
Type: Primary
Descriptions: The systems administrator runs and logs in to the Elections Administrator. He or she then uses this program to enter the minimum number of elections officers needed to make changes to the election and allows the elections officer(s) to enter their identification information and the election code of the new election.

Use Case: **Suspend Election**
Actors: System Administrator, Elections Officers
Type: Secondary
Descriptions: This use case occurs if an emergency such as a security breech occurs or if an election gets called off. The systems administrator runs and logs into the Elections Administrator. The minimum number of elections officers needed to make changes then enter their usernames and passwords. The election is then suspended, with no more votes or other changes to the database allowed. An e-mail is automatically sent out to all voters and elections officers in the election to inform them of this development.

Use Case: **Restore Election**
Actors: System Administrator, Elections Officers
Type: Secondary
Descriptions: This use case occurs if an election was suspended deliberately or due to some unforseen circumstance as a power failure, and the elections authorities wish to get it back into operation. The systems administrator runs and logs in to the Elections Administrator. The minimum number of elections officers needed to make changes then enter their usernames and passwords. The election is then restarted, with the settings restored to the state previous to the suspension of the election. An e-mail is automatically sent out to all voters and elections officers in the election to inform them of this development.

Use Case: **Set Election Settings**
Actors: System Administrator, Elections Officers
Type: Primary
Descriptions: This use case occurs when an elections officer wishes to configure a new election or edit an existing election before it has started. The minimum number

of elections officers needed to make a change to the system run and login to the Elections Editor and enter the election settings, ballot and enumeration list of authorized voters.

Use Case: **Obtain Voting Client**
Actors: Voter
Type: Primary
Descriptions: The individuals authorized to vote must either be provided with a version of the Pericles Voting Client by the elections authorities or they must be given access to computers running the software. The Voting Client will be included with the rest of the Pericles Election Software with an installation package that will allow it to be easily installed on computers.

Use Case: **Receive Voting Instructions**
Actors: Voter
Type: Primary
Descriptions: The default will be for voters to automatically be sent an e-mail by the Pericles Elections Server giving them all the information that they need to log in and vote in the appropriate election. This option may be disabled by elections officers, in which case the elections authorities will be required to provide the appropriate information to voters.

Use Case: **Ask For Help**
Actors: Voter, Elections Officer
Type: Secondary
Descriptions: This use case will occur when a voter encounters a problem that he or she is unable to solve independently. The voter will then have the option of writing an e-mail to the elections contact, whose e-mail address was distributed along with the e-mail providing voters with login information. It will then be the responsibility of this contact, usually an elections officer or officers, to respond.

Use Case: **Vote**
Actors: Voter
Type: Primary
Descriptions: The voter will run the Pericles Voting Client. The voter will then enter his or her login information, election code and the address of the Elections Server. This information will be sent to the Elections Server, after which the Voting Client will either be sent a ballot to fill out or a message explaining why the connection was unsuccessful. The voter will then send the completed ballot to the election server, after which he or she will be sent back a verification number to confirm that the vote was successful or a message saying that the vote could not be recorded.

Use Case: **Change Vote**
Actors: Voter
Type: Secondary
Descriptions: This will only be allowed in some elections, as decided by the elections officer(s). The voter will proceed exactly as under the Vote use case above. If the voter is not permitted to change his or her ballot then the voter will be sent back a message saying why.

IV. CLASS SPECIFICATIONS

The reader should note that the following class diagrams and class descriptions do not include the GUI information. This is because the system is designed to keep the functionality described in the class structures independent from the implementation of the GUI. See Appendix I for a description of how the various GUIs will operate.

4.1 Server System Classes

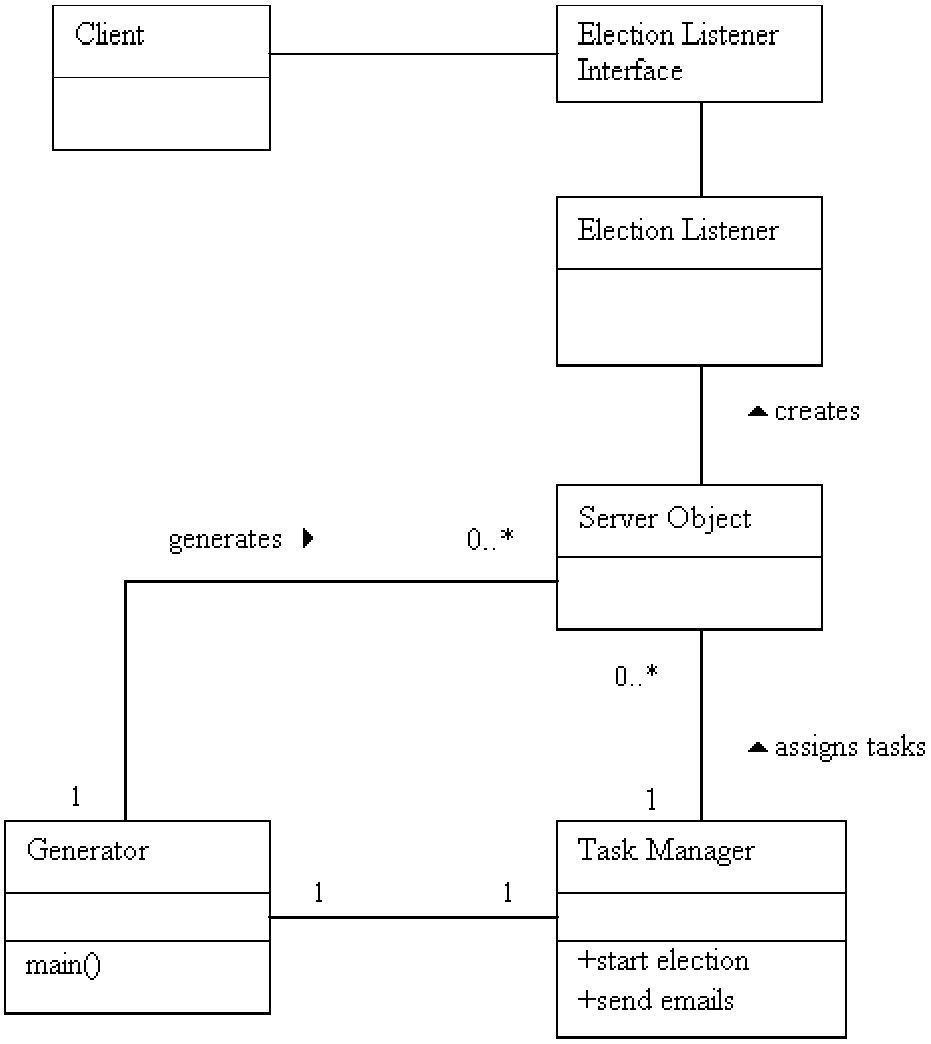

Figure 4.1: Server Overview

The Generator will be responsible for creating the individual server objects and the associated Election Object for each election. A Task Manager will hold a list of tasks for the Server Objects to perform. There will be one Task Manager and one Generator that oversee all elections, however each election will have its own Server Object and Election Listener. The Generator checks the database for new elections and tasks to perform on a regular basis.

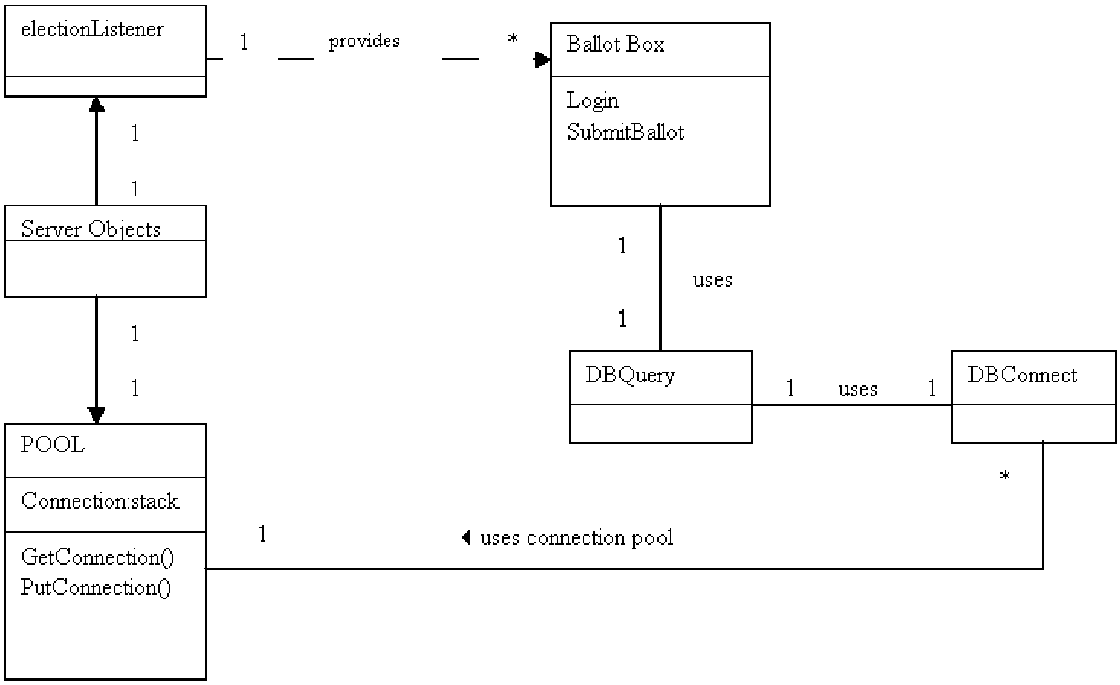

Figure 4.2: Server-Database Diagram

The Server Objects will hold a POOL of connections to the database. Whenever a Ballot Box needs to query or write to the database it will request a connection from the pool and release that connection back to the pool when finished with it. Because of this, Objects do not need to connect and disconnect from the database every time they need to access it. This should help performance. Note DBQuery uses DBConnect Object for any database connections. The DBConnect class grabs connections to the database through the Connection Pool, which resides as part of the Server Object.

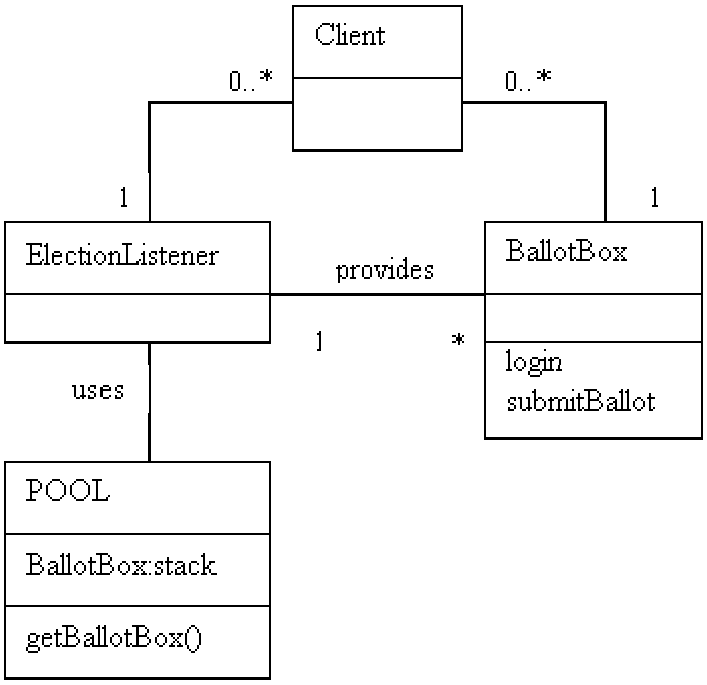

Figure 4.3: Client-Server Diagram

This diagram demonstrates the Client connection to the Server. The voter always connects to one ElectionListener object on the server. From this object, the client receives a BallotBox object which can fully service the client for the entire session. The ElectionListener pools BallotBox objects so that when server load is high, one BallotBox object is not responsible for servicing all Client systems.

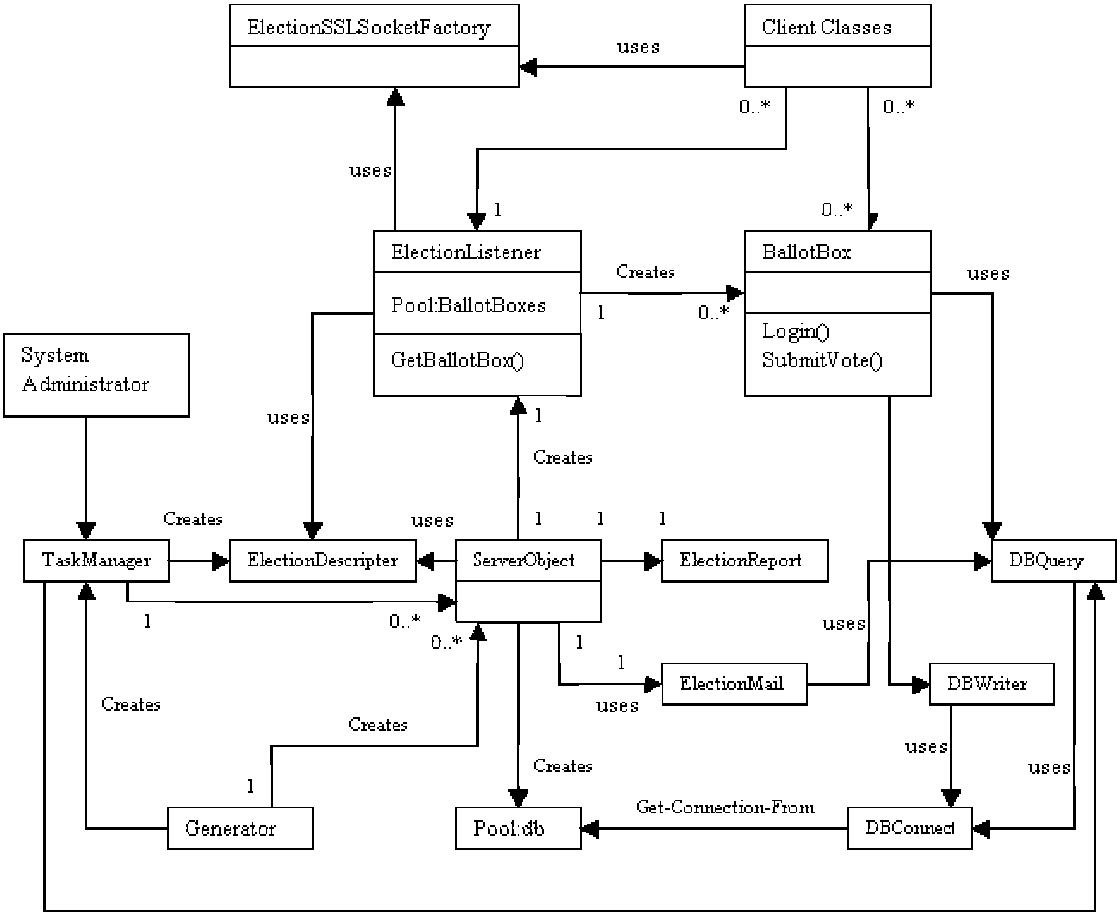

Figure 4.4: Complete Server Diagram

The digram represents the whole picture of the server model. This model demonstrates the objectives already described in the previous figures. It shows how the ServerObject pools database connections and the ElectionListener pools the Ballotbox objects. Added into this picture is how the Task Manager creates Election Descriptions for the ServerObject to use. These Election Descriptions have all the information about where the ElectionListener is to be registered with RMI and the connection information for the database. This enables a ServerObject to connect to any database. Also, the ServerObject functions to produce ElectionReports or ElectionEmails whenever the TaskManager assigns tasks to the ServerObject. The ElectionSSLSOcketfactory is also introduced in this diagram It is used by the Client Classes and the ElectionListener to establish secure transmission of data between the client to server through RMI. The System Administrator was also added to the diagram to show that he or she can add tasks for the Task Manager to perform by using the Elections Administrator.

Election Generator
The Election Generator is responsible for instantiating new elections and running a TaskManager which does periodic checking of the elections. The Election Generator also queries the database on a regular basis for new or updated information about elections.

Method Name	Formal Parameters	Return Values	Actions or Algorthim
Main			Starts all of the server objects that are required for each election and runs a thread for the TaskManager to do checks on these objects
DoTaskManager			Spawns off a TaskManager object which will perform system monitoring on a regular basis.

TaskManager
The task manager runs on a timer and is a RMI server to the Administrator program. If any election needs to be suspended or restored, the task manager assigns the responsibility to the appropriate Server Object. The task manager also regularly checks the status of elections. If any election is not available for connection but should be, the task manager emails the elections officer(s) to inform them that an election is not running. It also analyzes the election reports to see how many voters each server has served. If the amount is extremely high or volumes have changed drastically then an email is sent to the system administrator advising him/her to look at the log files (described in the appendix) for potential attacks.

Method Name	Formal Parameters	Return Values	Actions or Algorthim
GetAnyNewElections		Vector of Strings	Checks the database to see if there are any new elections that need to be created If any are created, a set of election codes are returned
CheckOnEachElection		reportObject	From all the elections that are created, check to see that they are active and get a report form each of them
CheckAnyStopElection			Sees if any elections need to be suspended
RestoreElection	String Election Code		Creates the ServerObject for the election and sets the ElectionDescription for the server object. Adds the server object to its regular task schedule.
HaltElection	String Election Code		Passes the command on to the appropriate

			ServerObject depending on the election code.

ElectionListenerImpl

The election listener hosts the Pool of BallotBox objects. When a client searches the RMI registry for a connection to the election, it connects to the election listener. The election listener passes the client a BallotBox object from the pool. This should speed up interaction, as all clients will not be using the services of the same object for every task. The election listener will pool BallotBox objects as well as db connections for the BallotBox objects. The client may take time to fill in the ballot which will tie up connections to the db, which is why they are pooled.

Method Name	**Formal Parameters**	**Return Values**	**Actions or Algorthim**
SetAllowVoting	Boolean	None	
SetElectionDescription	ElectionDescription	None	Sets the object that describes the Election Description
SendEmail	Enum email type	None	Sends email messages depending on the type of email message that is required.
StopElection	None	Boolean	Makes sure that all Ballotbox objects in the Pool are destroyed.
Constructor			Creates pool of BallotBox and ServerSQL objects.
GetBallotBox	None	BallotBox, or Null	Returns Null until ElectionOn is true.
GetAllowVoting	None	Boolean	
GetElectionDescription	None	ElectionDescription	
SetListenToOn	Boolean	None	Flag as to whether or not the election is on or not.

BallotBoxImpl

The BallotBox class represents the server object that the client classes will use to login, receive a ballot, and submit a ballot to be counted. The ballot box will return a verification number to the client classes. The ballot box will be an RMI object.

Method Name	**Formal Parameters**	**Return Values**	**Actions or Algorthim**
Login	Login Object	Ballot XML file	Calls DBQuerry.Login to see if the voter is legit for the election.
private FetchBallot	Ballot Type	Ballot XML file	If the login was legit, the ballot type is legit, if the login was not legit, the Ballot Type will be "could not login"
SubmitBallot	Ballot XML file	String	Verification number

Finalize			If client times out, BallotBox puts itself into the pool of BallotBoxes

ElectionReport

These attributes are meant to tell the Generator how the workload is and if there are any sudden bursts of activity. The ClientsServed is the number of clients served since the last time report was generated. The Generator will use this report to detect any high volume rushes.

Method Name	Formal Parameters	Return Values	Actions or Algorthim
setElectionCode	String		
setBallotBoxCount	Integer		
setClientsServed	Integer		
setConnectionCount	Integer		
getElectionCode		String	
getBallotBoxCount		Integer	
getClientsServed		Integer	
getConnectionCount		Integer	

ServerObject

The server object knows everything about a particular election. It ensures tasks get done that the ElectionGenerator requires to get done.

Method Name	Formal Parameters	Return Values	Actions or Algorthim
getElectionReport		ElectionReport	Returns an election report that gathers all of the information from the ElectionListener about the ongoing election
sendElectionEmails		Boolean	Creates an object that sends all of the election emails that will be necessary for the given election
getElectionResults		ElectionResult	Creates an ElectionResult Object that gathers all of the election results and returns it to the ServerObject
getDatabasePool		Pool Object	Returns a handle to the connection pool
setElectionDescriber	ElectionDescription		Stores the necessary election information locally
HaltElection			Destroys all of the election objects associated with the server Object. Removes the ElectionListener from the RMI Registry.

ElectionDescription

Describes all of the election-specific details for any particular election. This information is required by the ServerObject so that it can create database connections and register the ElectionListener with the RMI registry

Method Name	Formal Parameters	Return Values	Actions or Algorthim
SetDBURL	String		for db connections
SetDBHost	String		for db connections
SetDBPort	String		for db connections
setDBDriver	String		for db connections
setElectionCode	String		For the naming service
SetRMIComputerName	String		For the naming service
SetRMIPort	String		For the naming service
setServerLog	String		Sets location for the server log file.

Pool

Contains a stack of objects that can be pooled or re-used rather than destroyed. This Abstract Class will be used to pool BallotBox objects that serve clients connecting to the server and for database Connections. Any class that is pooled must implement the Poolable interface.

Method Name	Formal Parameters	Return Values	Actions or Algorthim
Constructor()		None	
Get		Object that is pooled	
Set		Object that is pooled	
SetMaximum	Integer		Maximum amount of objects at any one time

Interface Poolable

Before pooling an object, its class must implement poolable.

Method Name	Formal Parameters	Return Values	Actions or Algorthim
ObjectCreate		None	
ObjectReset		None	Reset any data that needs to be cleared
ObjectExpire()		None	If the object has been in the pool for some time, expire it.
objectValidate()		None	Validate that the object is in proper form to be used.

4.2 Client System Classes

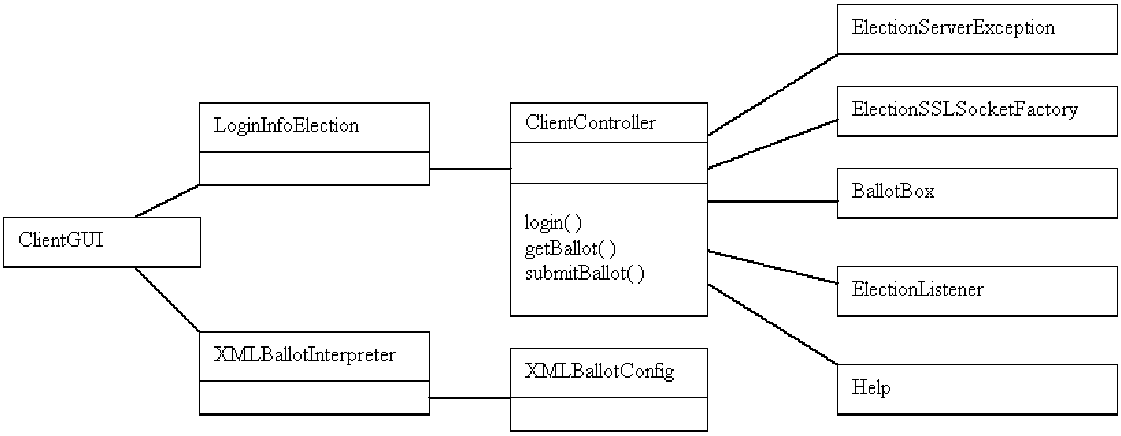

Figure 4.5: Client Diagram

The Client Classes for the Client System center around the ClientController class. When the Client GUI needs to Login a voter it creates a LoginInfoElection object and passes it to the ClientController to do the login. The client controller does all the RMI connecting and performs any server interactions. The Client GUI uses the XML BallotInterpreter when it needs to display a ballot. The ballot is stored in XML format.

ClientController
The ServerController is the way that the Client GUI interfaces with the server. The ServerController uses the classes ElectionServerException and ElectionSSLSocketFactory, and Help which are Shared classes.

Method Name	Formal Parameters	Return Values	Actions or Algorthim
Constructor			Creates the SocketFactory layer for communication to the server.
Login	Login Object	Ballot XML file	Calls the BallotBox to do this. The
GetBallot			Calls the BallotBox to do this
SubmitBallot	Ballot XML file	String	Verification number

BallotBox
The BallotBox class represents the server object that the client classes will use to login, receive a ballot, and submit a ballot to be counted. This ends up being an RMI stub on the client side. The ballot box will also return a verification number to the client classes. The ballot box will be an RMI object.

Method Name	Formal Parameters	Return Values	Actions or Algorthim
Login	Login Object	Ballot XML file	Calls DBQuerry.Login to see if the voter is legit for the election.
SubmitBallot	Ballot XML file	String	Verification number
Finalize			If client times out,

			BallotBox puts itself into the pool of BallotBoxes

ElectionListener
The ElectionListener class represents the server object that the client classes will use to make first contact with the server. After connecting through RMI, they will receive a reference to a BallotBox object that will serve them for the remainder of the connection session..

Method Name	Formal Parameters	Return Values	Actions or Algorthim
getBallotBox		A reference to a BallotBox.	Serves the Voting Client software a reference to a BallotBox Server object. The client can then use the BallotBox to participate in the election.

4.3 Election Editor System Classes

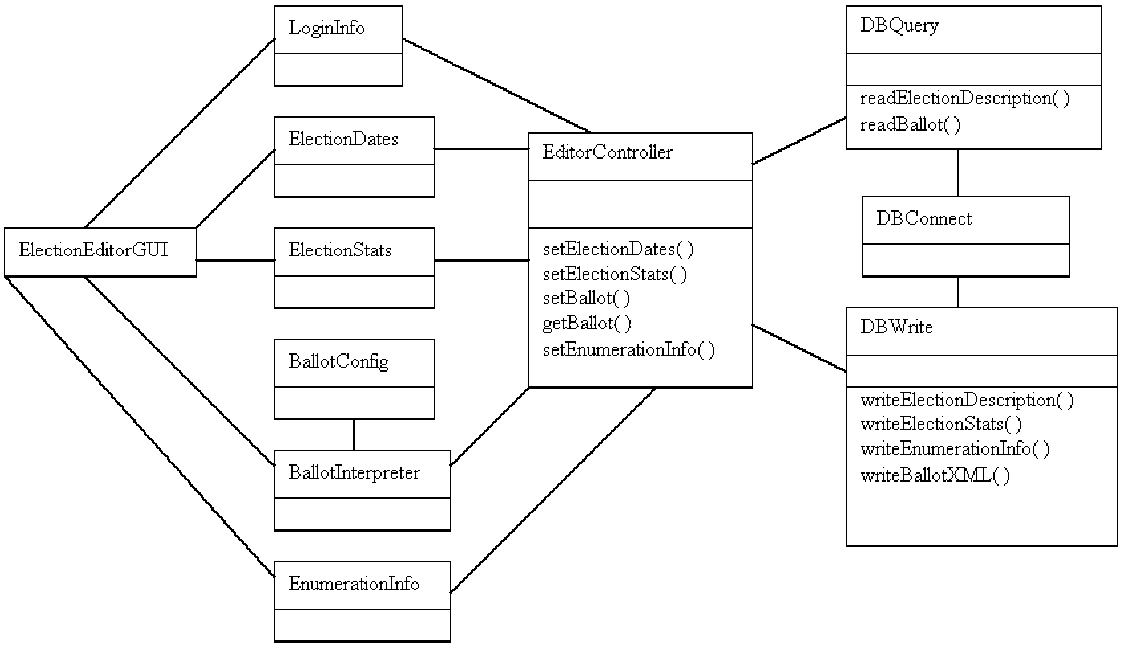

Figure 4.6: Election Editor Diagram

The election editor program is modeled similarly to the Client class, where an Editor controller class interfaces to the GUI to do any of the interactions with the backend. This application is not connected to the server in any way. The backend is a database. The DBQuery and DBWrite are used to store election preferences and ballot descriptions. The description of the look and feel of the ballot is stored in XML form in the database.

The only two objects specific to this system besides the GUI classes are the Editor Controller and the EnumInfo class. The rest of the classes are listed in the shared classes section.

EditorController
The EditorController is a controller class used to interface with objects that are created by the GUI to represent user input. These objects are passed along to the appropriate DB class.

Method Name	Formal Parameters	Return Values	Actions or Algorthim
Constructor			
SetElectionDates	ElectionDate		
SetElectionStats	ElectionStats		
GetElectionDates		ElectionDate	
GetElectionStats		ElectionStat	

EnumInfo
This class is sub-classed from loginInfo to store information about voters.

Method Name	Formal Parameters	Return Type	Actions or Algorthim
SetEmail	String		
SetWeighting	Integer		
GetEmail		String	
GetWeighting		Integer	

4.4 Election Administrator System Classes

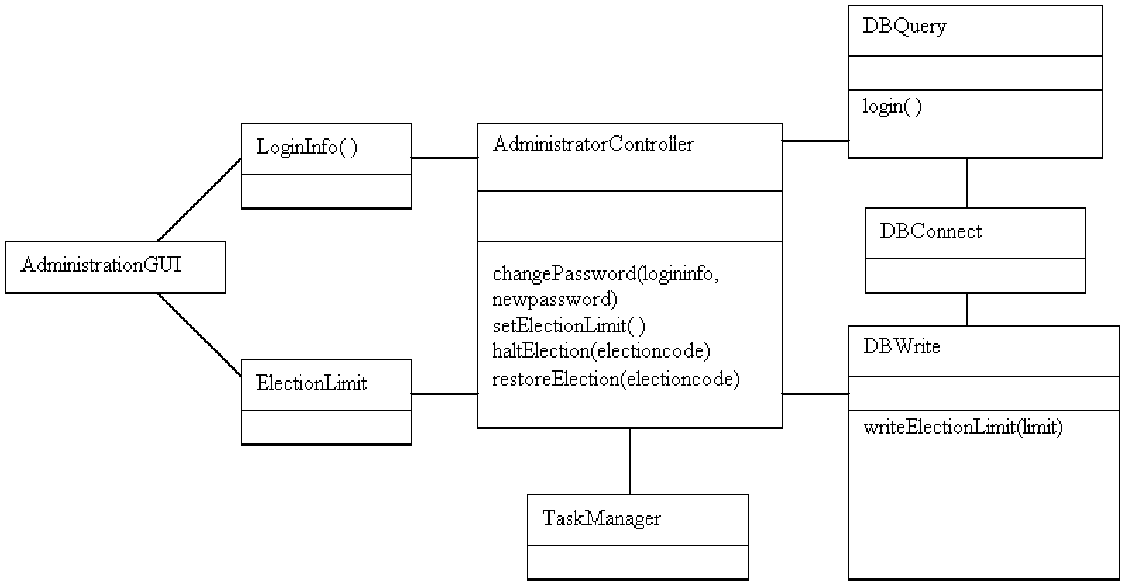

Figure 4.7: Administration System Diagram

Modeled similarly to the ElectionEditor, the Election Administer application has an AdministrationController which interfaces with the GUI and to the Database classes

AdminController:
This class will be used by System Administrator to authorize election officers to register elections. It will Deal with security issues during elections and suspend and restart an election that was security breached.

Method Name	Formal Parameters	Return Type	Actions or Algorthim
ElectionAdmin			Constructor
ChangePasswd	loginInfo password new	Boolean	Change system administrator's passwd
RestoreElection	int electCode	Boolean	Makes call to the Task Manager to restore election
HaltElection	int electCode	Boolean	Makes call to the Task Manager to halt the election
SetElectionLimit	ElectionLimit limit	Void	
BreachAlarm	int electCode	Void	
RegisterOfficers	registeredOfficer	Void	Calls db class to add all the election officers to the Elections Officers table and the int minimum to the election table.

ElectionLimits:
This class includes information about limitations that the system administrator imposes on the system.

Method Name	Formal Parameters	Return Type	Actions or Algorthim
ElectionLimits			Constructor
ElectionLimits	int electTime int dataTime int numVoter int numElect		Constructor
SetElectTimeSpan	int electTime	Void	
GetElectTimeSpan		Int	
SetDataSaveTime	int dataTime	Void	
GetDataSaveTime		int	
SetMaxNumVoter	int numVoter	Void	
GetMaxNumVoter		Int	
SetMaxNumElect	int numElect	Void	
GetMaxNumElect		Int	

RegisteredOfficers:
This class will be used by the system administrator to authorize election officers to register election information.

Method Name	Formal Parameters	Return Type	Actions or Algorthim
RegisterOfficers			Constructor
SetElectCode	int electCode	Void	
SetOfficers	vector loginInfoElec	Void	
SetMinOfficer	int min	Void	

4.5 Database Classes

The database has been broken into three classes. One is a connection class that holds the information required to connect to the database and the connection objects to the database. The other two classes hold the SQL to do writes and queries on the database.

DBConnection
This class is a container for holding a connection to the database. The database can therefore be changed at will. The ServerObject class uses the ElectionDescription to get the database settings. The DBConnections are pooled by the ServerObject.

Method Name	Formal Parameters	Return Values	Actions or Algorthim
Constructor()			
Constructor()	string pass, string login, string url, string driver, string server		
SetPassword	String		
SetLogin	String		
set url	String		
set driver	String		
set Server	String		
GetConnection		Connection	Returns a connection to the database
GetConnection		Connection	returns a connection to the database
ConnectDB			does the connection to the database

DBError
The class extends exception

Method Name	Formal Parameters	Return Values	Actions or Algorthim
Constructor()	String Error	None	

DBQuery
The class that contains all of the querries needed for getting results from the database

Method Name	Formal Parameters	Return Values	Actions or Algorthim
Constructor	Connection		
DBLogin	Login Object	Boolean	Use the Login object to validate that the user is valid for the election code of the project.
Private getConnection		Connection	Gets a Connection from the DBPool to use for the query.
GetBallot	BallotType	XML file	

DBWrite
The class that is used to write to the database. This classs uses DBConnect to write.

Method Name	Formal Parameters	Return Values	Actions or Algorthim
Constructor()			

WriteElectionDates	ElectionDate object, election code		
WriteElectionLimit	ElectionLimit		Stores all of the election limitations set by the administrator

4.6 Shared Classes

The following classes are used by more than one section of the system.

Help

System classes can use this static class to display help.

Method Name	Formal Parameters	Return Values	Actions or Algorthim
DisplayHelp	String name of wizard window	File for appropriate help	

LoginInfo

Used to store login information. This class will work for the administrator only. To login an election officer or a voter, the extended LoginInfElection can be used.

Method Name	Formal Parameters	Return Type	Actions or Algorthim
LoginInfo			Constructor
LoginInfo	String id String passwd		Constructor
SetID	String id	Void	
GetID		String	
SetPasswd	String passwd	Void	
GetPasswd		String	

LoginInfoElection

This class extends the super LoginInfo, which allows voters and elections officers to login to our system.

Method Name	Formal Parameters	Return Type	Actions or Algorthim
setElectCode	int electCode	Void	
getElectCode		Int	

DataGenerator

This is a static class that will generate random string for user login ID and password and a random number for voting verification number.

Method Name	Formal Parameters	Return Type	Actions or Algorthim
GetID	int key	String ID	
GetPasswd	int key	String passwd	
getVerifyNum	int key	int number	

ElectionStats

This class is used to encapsulate statistic information about a particular election. This class holds what statistics are to be reported about an election and carries the task of generating the statistics and putting them in the appropriate place.

Method Name	Formal Parameters	Return Type	Actions or Algorthim
- getNumQuestion	int electCode	Int	
- getVoteCount	int choice	Int	
- getVoteResult	String filename	Void	
getDoDisplayWinners		Boolean	Stored in db
setDoDisplayWinners	Boolean		Stored in db
getDoDisplayNumbers		Boolean	Stored in db
setDoDisplayNumbers	Boolean		Stored in db
getDoDisplayPrecentages		Boolean	Stored in db
setDoDisplayPercentages	Boolean		Stored in db
doElectionStatsReport			carries out the report process of generating the statistics. This interfaces with DBQuery to get any results it needs.
parseToHtml			parses the results to HTML format.
postReport			
setLocationGeneral	string location		
setLocationComplete	string location		
setLocationVoters	string location		

ElectionDates

This class encapsulates all of the dates that are needed for a particular election. It is used by the Election Editor to get dates from the GUI to the database.

Method Name	Formal Parameters	Return Type	Actions or Algorthim
setEmailSend	datetime		
setElectionStart	datetime		
setElectionEnd	datetime		
setElectionPublish	datetime		
setRemovalDate	datetime		
setTimeWindow	datetime		
setTimeZone	enum		
getEmailSend	datetime		
getElectionStart	datetime		
getElectionEnd	datetime		
getElectionPublish	datetime		
getRemovalDate	datetime		

SecureEncryption

Method Name	Formal Parameters	Return Type	Actions or Algorthim
SecureEncryption			Constructor
SecureEncryption	int key		Constructor

SetKey	int key	Void	
GetKey		int key	
encryption	String filename	Void	
decription	String filename	Void	

ElectionSSLSocketFactory
This class is a sub-class of the Java RMISocketFactory class.

Method Name	Formal Parameters	Return Type	Actions or Algorthim
createSocket	String host int port	Socket	throws IOException
createServerSocket	int port	ServerSocket	throws IOException

ElectionException
This is a collection of exception classes defined in our Pericles system. All these exception classes extend Exception class.

Exception Class Name
IDNotFoundException
PasswdNotMatchException
ElectionCodeNotFoundException
BallotNotFoundException
TimeOutException
LateVoteException

ElectionEmail
This is a class that can be used to send emails to voters, election officers or system administrators.

Method Name	Formal Parameters	Return Type	Actions or Algorthim
Constructor	Vector address Integer type		
Constructor			
SetEmailType	Integer type		
SendMail			parses through all of the email addresses in the vector and sends the email of the message type

XMLBallotConfig:
This is a utility class that will handle ballot configuration information. Loads and parses ballots written in XML.

Method Name	Formal Parameters	Return Type	Actions or Algorthim
BallotConfig	String filename		throws IOException
BallotConfig	InputStream in		throws IOException
GetDriverClass		String	
SetDriverClass	String driverClass	Void	
GetHandlers		Hashtable	
SetHandler	Hashtable handler	Void	

ParseConfiguration		Void	throws IOException
SaveConfiguration	String filename	Void	throws IOException
SaveConfiguration	OutputStream	Void	throws IOException

XMLBallotInterpreter
This class will handle the XML ballots and display them to the GUI.

Method Name	Formal Parameters	Return Type	Actions or Algorthim
DisplyBallot	String filename		display ballot config file to GUI throws IOException
SaveBallot	String filename		save ballot configuration as XML file throws IOException

V. COLLABORATIONS

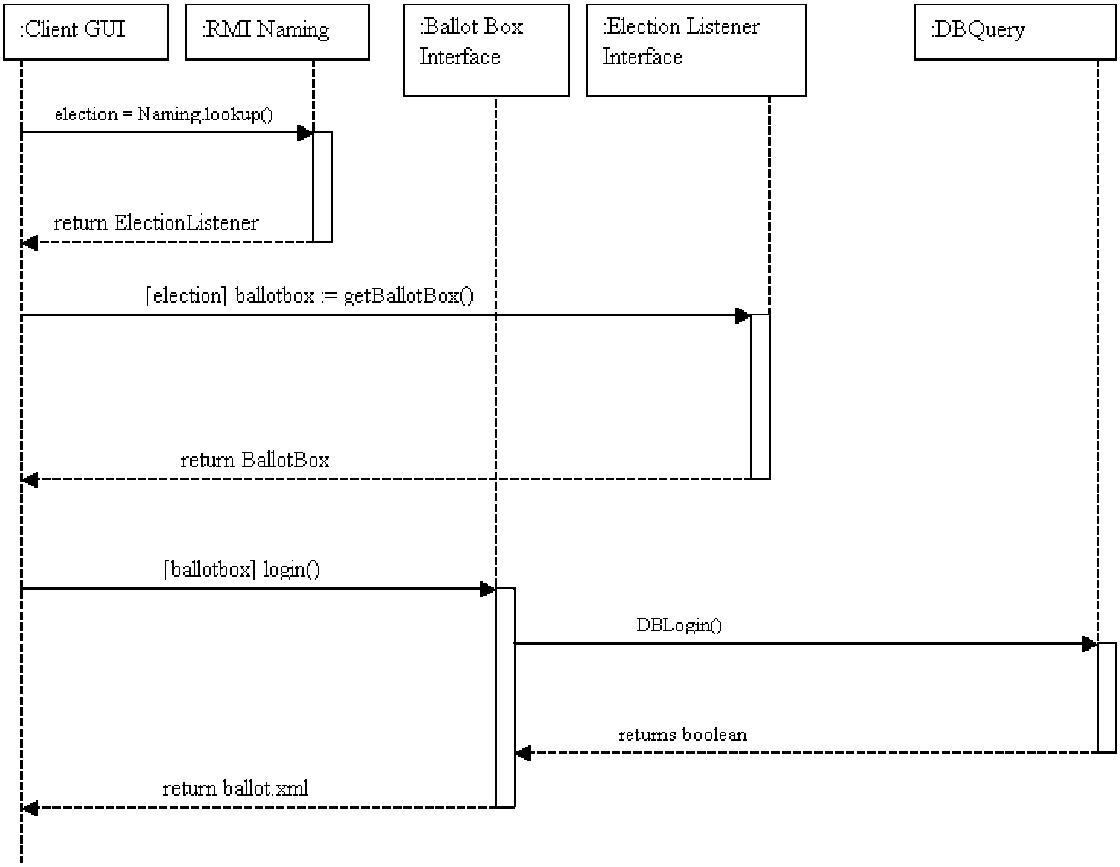

Figure 5.1: Client Logs on and Receives Ballot

The Client Controller gets a remote reference to an ElectionListener by doing a lookup in the RMI registry. It then gets a BallotBox reference remote object to do the login to get a ballot description in XML format.

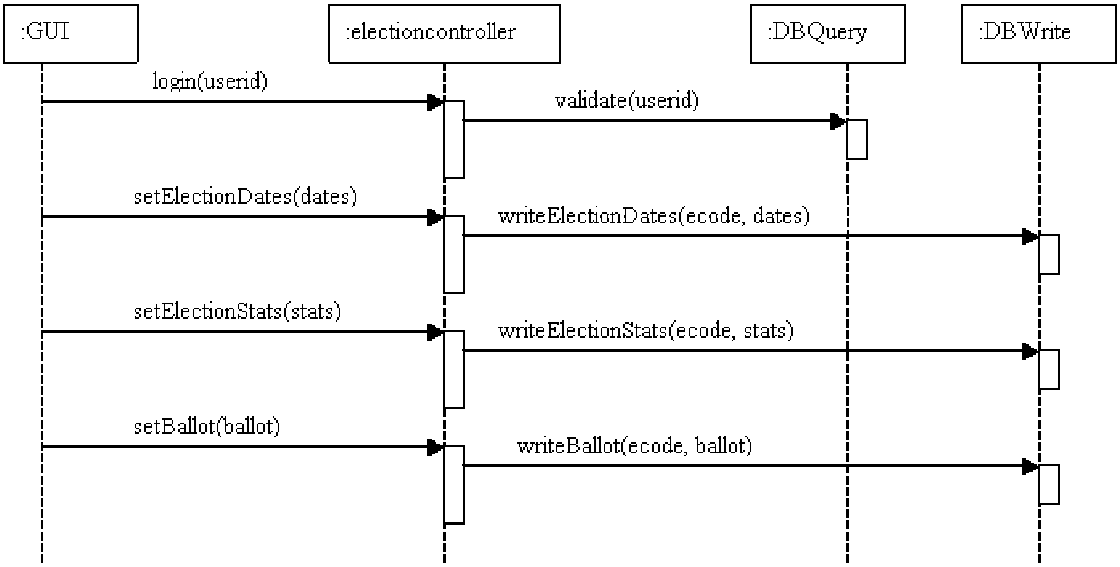

Figure 5.2: Creation of an Election

Figure 5.2 demonstrates the role of the ElectionController in interfacing with the database on behalf of the GUI classes. First the elections officer must login, then set all the dates for the election, the stats to be collected and then the description for the ballot which is parsed in XML format.

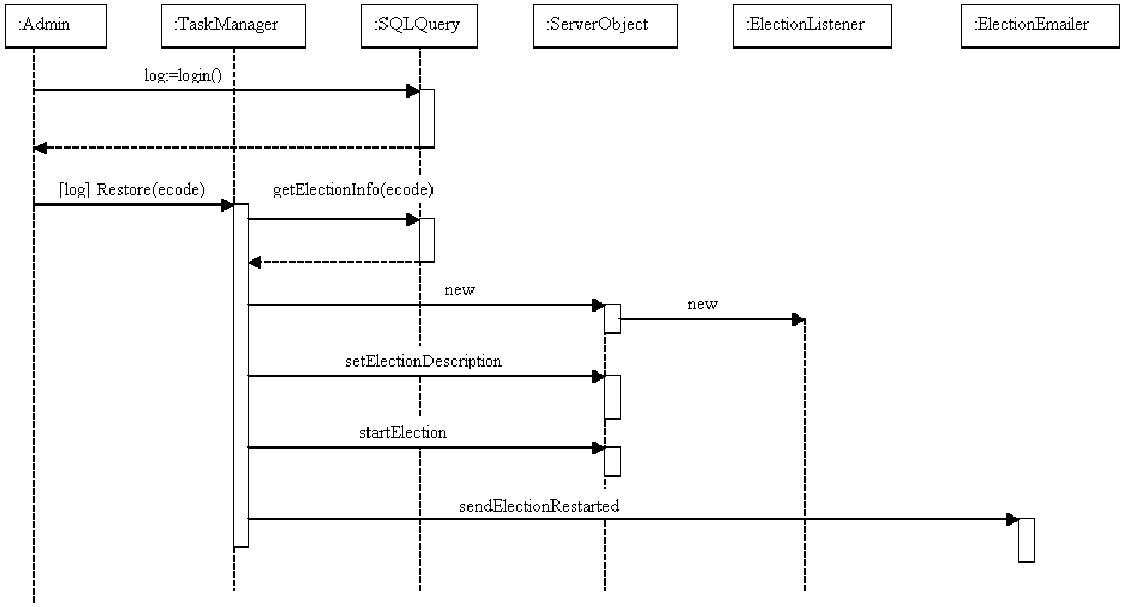

Figure 5.3: Restarting an Election

This diagram demonstrates the series of events that occur when an administrator restarts an election. The administrator must login, and then put in the election code of the election to restart. The task manager creates a new serverObject and sets the description of the election on the ServerObject. The ServerObject then creates the ElectionListener for the clients to connect to.

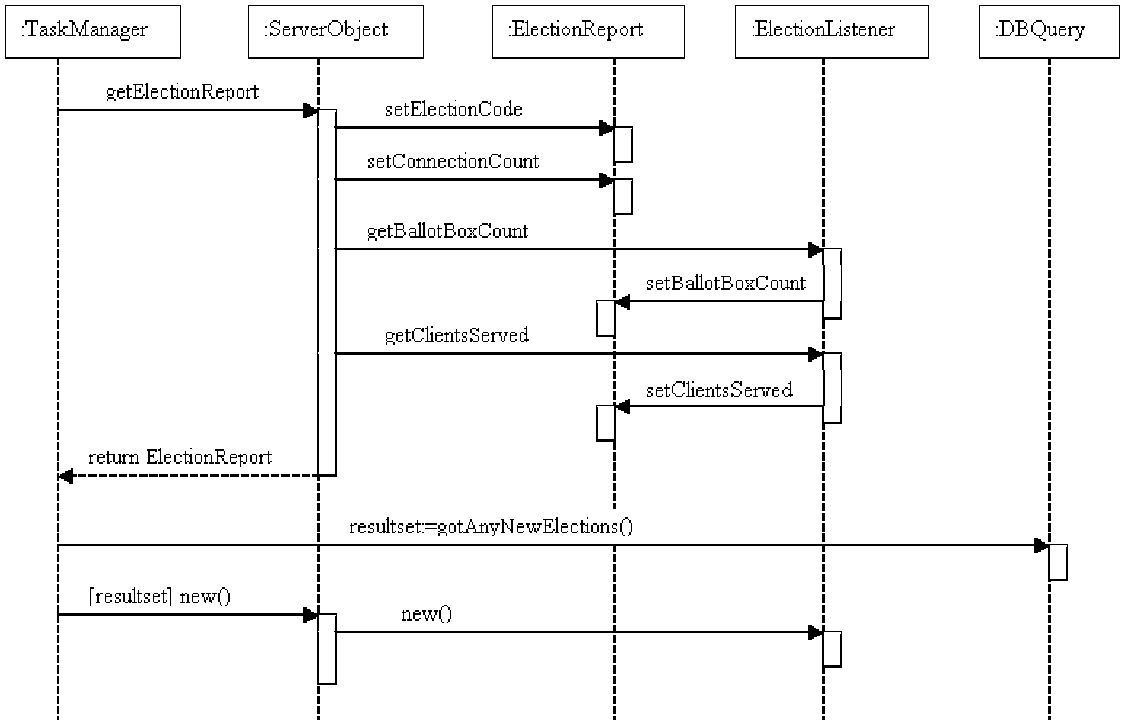

Figure 5.4: Checking on Election Servers

This diagram shows a typical TaskManager check of the Election Servers on the system. It gets an ElectionReport from each of the ServerObjects to process and add to a log file. It then gets any new elections registered into the database by the Election Editor. It creates a new ServerObject for the election if it finds a new election.

VI. CLASS HIERARCHIES

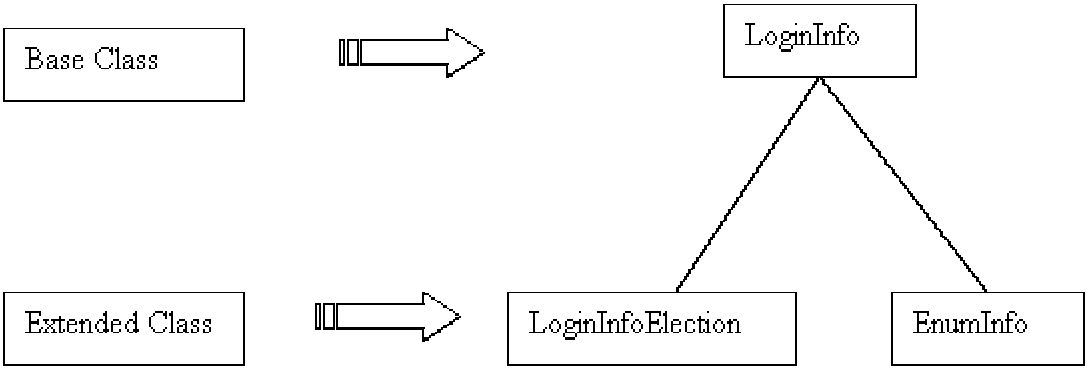

Figure 6.1: Login Information Inheritance

The LoginInfo class is the base class for other login classes that are used for enumeration and for all voter, election officer, and system administrator authentication

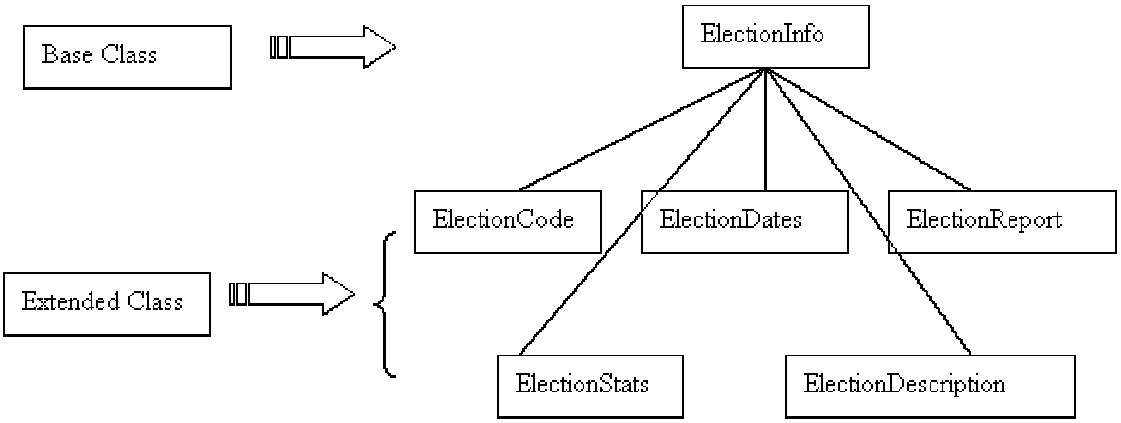

Figure 6.2: Election Information Inheritance

The base class ElectionInfo ensures that all Extended classes hold election codes. These classes all hold information about a particular election in the system. The Database Classes use these classes to write and read from the database.

VII. JUSTIFICATION

The primary priorities of this design are, in order of importance:

1) Functionality: the first priority is making sure that the system actually works. Without functionality, the system would be useless.
2) Extensibility: important because the primary goal of the first release of the Pericles system is to provide a base upon which more sophisticated systems can be built.
3) Security: nobody must be able to record unauthorized votes or interfere with the proper operation of the system.
4) Privacy: it is essential that nobody be able to know how any individual voters voted.
5) Scalability: it is desirable that the system could be easily scaled up to handle major elections.

In order to accomplish these priorities, every effort was made to incorporate modularity, information hiding, low coupling, high cohesion and data encapsulation into the design. Not only are these important characteristics of object oriented programming, but they are also particularly important in achieving an easily maintainable and secure system. The sections below highlight some of the ways in which the design implements these five priorities:

7.1 System Administrators and Elections Officers

These two types of actors were assigned their roles in order to help guard against too much power being placed in the hands of too few potentially corruptible elections officials. Preventing the systems administrator from having access to any of the functionality of the Elections Editor prevents him or her from having the power to corrupt ballots, influence enumeration lists or make changes to elections settings. Preventing elections officers from having access to elections other than their own limits the danger of their corruption, as does the option of requiring more than one elections officer to make changes. Both elections officers and systems administrators are required to log in in order to suspend or restart elections because of the impacts of these actions, both legally and on the computer systems running the Election Server. The differences between the roles of these two types of agents are reflected in the software design by the distinction between the Elections Administrator and the Elections Editor software.

7.2 Server Architecture

The Election Generator was assigned the role of task master for the Election Objects so that there would be one central object ensuring that everything is functioning as it should. This is essential for detecting technical failures or security breeches like denial of service attacks that would incapacitate the system. The log files can be used to track inappropriate uses of the system (see Appendix VI).

The decision of having one distinct Election Object per election was made so that a breech in security or an extraordinary load would only affect one election rather than all of the elections running on an Elections Server. Another essential benefit of this is that it allows for the use, in future versions of the software, of different Election Objects on different computers controlled by a single Election Generator housed on a single computer. This is important in terms of

scalability, since it helps to accommodate large elections by using the resources of multiple computers.

The use of separate objects to send e-mails, communicate with clients and record results is important because it allows for the kind of specialization that is important for the sake of extensibility.

SSL Socket Factories were used to provide encryption because they are the most advanced and reliable means available to the implementation team for ensuring transmission security.

7.3 Voting Client

The Voting Client was designed to be simple and easy to use. It was designed to be little more than a ballot reader so that it could accommodate a wide range of election types. It is thus very easily extensible, since the only likely changes that would need to be made to it are in the ballots descriptions and the associated XML tags. The Voting Client and the Elections Server also communicate error messages and other types of information through XML "ballots," which is also very beneficial in terms of extensibility, since it allows for a changing variety of communication types.

7.4 Database

The database schema is designed so that, even if somebody managed to gain direct access to it, they would not be able to link a voter's votes with his or her identity (as given by e-mail address). The ways in which data is separated in the tables also makes it easy for changes in design to be incorporated into single tables without impacting other tables.

The database schema was also designed to follow the rules of good relational database design. There is no redundancy other than that which is necessary for security reasons, and the tables are cleanly and logically divided. This makes the database easily extensible, as does the inclusion of key columns that will not be used in the current implementation (such as EmailID). These could be used by future designers to link tables in the database in ways that are not currently needed.

The system was also designed in such a manner that it would be easy to change it to accommodate databases that are on different computers than the server. This is very beneficial in terms of scalability, since it might be useful in large elections to use more than one database located in more than one place.

7.5 GUIs

All of the GUIs in the Pericles Election System were designed with a sequential wizard-type interface. This was done in order to ensure that no details got left out, since everything is essential in an election system. It also makes the system easily extensible by simply adding, removing or editing individual screens rather than the whole interfaces of the programs. All of the functionality of the programs is designed to be external to the GUI classes, in order to make the software more portable.

7.6 E-mail

Automatically generated e-mails were used extensively in the design for security reasons. By having the system randomly generate e-mails and send them out to voters, it not only increases security because there is no external access to identification information, but it also increases privacy by allowing the database schema to be designed in such a way as to isolate the information indicating the identity of voters from their votes.

E-mail also increases the extensibility of the system. If voters need to be informed of new information, it can simply be incorporated into new ballots and new e-mails without requiring any changes to the Voting Client or its associated help files.

The down side of the use of e-mails and randomly generated login information is that it restricts accessibility to those who have access to e-mails. In order to compensate for this, the design team has incorporated the option of not using e-mails and automatically generated login information, at the discretion of elections officers.

7.7 Configurability

The Pericles Election System has been designed to be highly configurable when designing elections, as is exhibited in section 3.3.1.2. This is essential for allowing the flexibility that is needed to encompass the wide range of elections that are possible.

VIII. EXTENSIBILITY AND SCALABILITY

Extensibility and scalability are two of the key strengths of the Pericles Elections System. It is designed so that changes can be easily made to certain portions of the design without requiring major modifications to other parts of the system.

This is especially important because of the experimental nature of the Pericles system. It is meant to be an exploration of the field of electronic voting, and as such will inevitably require post-production modifications and improvements. Scalability is essential because of the limited resources available to the testing team. Ideally, this system could eventually be scaled up to service full large-scale elections.

8.1 Server
The use of Server Objects and the ways in which the database is accessed are designed so as to need only small and isolated modifications to move the Server Objects and database onto different computers from the one which houses that Elections Generator. This would accommodate large elections by taking advantage of the resources of multiple computers. The configuration file of the Elections Server is can be easily modifiable to reflect any such changes.

8.2 Database
The database schema has been designed to have linking columns that are not currently needed and tables are separated in such a way as to logically separate information. This means that the new types of information can be added without needing to change more than a few tables. All communications between the Election Server and the database are passed through a database translator class, which means that this will be the only class that needs to be changed if a non-MySQL database is accessed by a future version of the software.

8.3 GUI
The use of wizard-type interfaces means that changes can be made to single screens of the interfaces without needing to change anything else.

8.4 E-Mail
Any changes to the information that needs to be communicated in the e-mail can be incorporated directly into the files that are used as e-mail templates without needing to change any compiled software.

8.5 Ballots
Any changes to the format and types of information that appear on ballots can be made directly to the ballots and the associated XML tags without needing to change the Voting Client. In addition, XML “ballots” are used to communicate error messages and other types of information between the Server and the Client so that the Pericles Elections System can be easily modified to exchange a greater diversity of material. The separation of the Elections Editor from the Elections Generator makes it possible to make these kinds of changes without needing to change the Elections Generator or its Server Objects.

8.6 Voter Identification Information
The option of using pre-existing usernames and passwords has been incorporated into the design. This has been done so that it will be easy to set up future versions of the software to download existing usernames and passwords from an external system. The database schema

has also been set up to isolate the information, for reasons of privacy, but the design of the schema has unused columns in it that could be used in future versions of the software to link isolated tables if so desired. This could be useful if, for example, it became desirable to have the ability to contact or trace individual voters.

8.7 Server Statistics

There is an unused table in the database called ServerStats. A great deal of information is also stored in log files (see Appendix VI). This will make it easy to change the software so that it will provide statistics on use patterns so that the systems administrators will have the ability to monitor and control network traffic.

IX. TEST PROVISIONS

9.1 Unit Testing

All modules of the code will be individually tested before being integrated with other parts of the system. This is to help ensure that errors in the code are caught at the earliest possible stage. Attempts will be made to test all possible paths of execution at this stage, although this becomes more and more difficult with large modules. At the very least, all boundary values will be tested.

9.2 Functional Testing

Functional testing will be conducted after the integration stage to ensure that no functionality has been corrupted during the integration process. These tests will include:

- Registering a new election and creating a ballot
- Voting in this election
- Querying the database to ensure that the votes were properly recorded
- Checking that the generated results statistics are correct.

9.3 System Stress Testing

Once a single election is working properly, the system will be tested with additional elections running simultaneously to ensure that running additional elections at the same time will not corrupt any data. This will also help to determine how many elections can be run simultaneously per system.

9.4 System Load Testing

For both single and multiple elections, system load testing will help ensure that the Election Servers can handle the traffic they are likely to receive during real elections. Scripts will be used to simulate multiple logons and votings during a short period of time so as to properly test the network capacity of the server and response time during peak load periods.

9.5 Data Verification Testing

9.5.1 Calculation of Election Results

Testing of the database will be done using a large sample of data to ensure that results of the election are properly calculated and reported. This testing will also attempt to overload the database to ensure that large volumes of data can be safely stored.

9.5.2 Dates

Tests will be applied to the boundary conditions of all dates specified during election setup. Tests will include:

a) Clients attempting to vote before start dates
b) Clients attempting to vote after end dates
c) Large numbers of clients attempting to vote close to the end of an election

9.6 Input Testing

All parts of the system will be tested with erroneous input to help guard against misuse and security breeches. The testing will also ensure that the appropriate informative error messages are generated.

9.6.1 Client Input Testing

This test will ensure that a given voter can only vote in elections that they are authorized for. Testing will also ensure that clients cannot alter their votes in an election where this is not permitted.

9.6.2 Election Generation Testing

These tests will ensure that an election cannot be registered with incorrect options, such as an end date before the start date.

9.7 Security Testing

The security of the elections will be tested in several ways:

9.7.1 Denial of Service Attacks

The testing team will attempt to shut down the servers using denial of service attacks and ensure that that the system can detect the attacks and take action.

9.7.2 Encryption Breaking

The testing team will attempt to intercept and decode the transmissions between the Voting Client and the Elections Server to ensure that encrypted data cannot be accessed within the running time of the election.

9.7.3 Database Breaches

The testing team will attempt to breach the security of the database and access or change the data contained therein. They must ensure that no information stored in the database can be accessed and that, even if it can be, that the schema prevents the ability to glean any useful information.

9.8 Recovery Testing

The testing team will kill the Elections Server in the middle of its operation in order to ensure that it can be restarted and that the contents of the database will remain up to date and secure. It will also ensure that a mailing goes out to contact voters and tell them that the server is back up.

X. EXTRA FEATURES & OPTIONAL FEATURES

10.1 Extra Features

A number of features beyond those in the basic requirements given to the design team by the client have been included in this design in order to make the software more functional, flexible and secure. The highlights of these are as follows:

1) Ability to accommodate elections with weighted votes
2) High level of configurability
3) The contacting of voters by e-mail to inform them of their registration in elections or if an election is halted or restored in order to increase security
4) Allows more than one level of security
5) The system administrator has the ability to impose global limits on elections
6) The separation between systems administrators and elections officers and the ability to require a minimum number of elections officers to make changes gives an added increase in security.
7) Allows both refused ballots and spoiled votes

10.2 Optional Features

The following are extra features that have not been incorporated into this design, but which may be implemented in this release if there is time:

1) Elections officers and/or voters able to tailor the appearance and style of the ballot
2) The keeping of network traffic statistics
3) User verification numbers can be used to trace the votes of given voters if they request verification of how they voted.
4) Allow for ballots with more than one answer allowed per question
5) Allow for ballots with questions that depend on the answers to other questions
6) Allow for ballots that permit voters to enter strings of text
7) Allow Server Objects to be instantiated on computers other than the one running the Election Generator
8) Allow the Election Generator to access databases on external computers
9) Automatically FTP the web pages containing elections results to other servers
10) Allow voting to be done through e-mail rather than with the Voting Client
11) Give printing functionality to the Voting Client
12) Create graphical representations of election results
13) Produce printed copies of election results
14) Produce election results in formats other than HTML
15) Place requirements such as a quorum to make results binding

XI. GLOSSARY & REFERENCES

11.1 Glossary

ASCII: a standard 7-bit code for representing characters—letters, digits, punctuation marks and control instructions—with binary values, the code values ranging from 1 to 127.

Authorized Voter: a voter who is on the enumeration list and who has been given the means to use the Pericles Voting Client software.

Ballot: an XML file transferred between the Voting Client and the Elections Generator.

Client: a program that connects to a server. In the case of the Pericles system, the Voting Client is a piece of software that Voters can use to connect to a Pericles Election Server and receive ballots, send their votes and possibly change their votes if this is permitted in an election.

CORBA: acronym for Common Object Request Broker Architecture. It is a distributed object-computing infrastructure that automates many common network-programming tasks.

Customers: peoples or agencies who purchase our Pericles system.

Database: any aggregation data. Files consisting of records (or tables), each of which is constructed of fields (or columns) of a particular type, together with a collection of operations.

Denial of Service: explicit attempts by attackers to prevent legitimate users of a service from using that service. Examples include:

- attempts to "flood" a network, thereby preventing legitimate network traffic
- attempts to disrupt connections between two machines, thereby preventing access to a service
- attempts to disrupt service to a specific system or person

Development Team: a team of people developing software.

Distributed Network: a network in which processing, storage and other functions are handled by separate units rather than by a single main computer.

Elections Administrator: GUI based application that allows systems administrators to authorized elections, set global limitations on elections, suspend elections and restore elections, sometimes in conjunction with elections officer(s).

Elections Database: a MySQL database system used for storing the information used in an election.

Elections Editor: Graphical User Interface (GUI) application that allows Election Officers to change election settings, customize ballots and enter enumeration lists.

Elections Generator: a runnable program that creates Election Objects and works as a task master for them.

Elections Objects: objects created by the Elections Generator that handle all communication with Voting Clients.

Election Officer: an impartial individual authorized to run an election.

Elections Server: all of the Pericles software on the server side. Includes the Elections Administrator, the Elections Editor, database interface objects and the Elections Generator (and the objects spawned from it).

Encryption: the process to making information indecipherable to unauthorized readers. This can be used for the storage and transmission of information.

Enumeration List: a list of identification information and other required information for all authorized voters.

Enumerated Voter: a voter that is registered for specific election.

Expired Election: an election that has come to a completion.

GUI: a graphics–based user interface that incorporates icons, buttons, pull-sown menus and a mouse.

HTML: an acronym for HyperText Markup Language. The document format used on the World Wide Web. Web pages are built with HTML tags, or codes, embedded in the text.

Linux: a version of Unix.

Log: a record of activities that take place on a computer system.

MySQL: a relational database management system (RDBMS) that uses Structured Query Language (SQL) for adding, accessing and processing data in a database.

Ongoing Election: an election that is currently in progress.

On-line Help System: a reference on how to use a piece of software that is integrated into the software.

Operating System: the software responsible for controlling the allocation and usage of hardware resources such as memory, central processing unit (CPU) time, disk space and peripheral devices.

Packet Sniffing: the act of intercepting information traveling across a network.

Registered Election: an election that has been registered with a Pericles Ballot Generator by a systems administrator and elections officer(s).

RMI: Acronym for Remote Method Invocation. It is a remote procedure call (RPC), which allows Java objects stored in a network to be run remotely.

Security Breach: the act or a result of breaking the security of a software system.

Server: a program that serves clients and controls access to all or part of a computer's resources.

System Administrator: a person responsible for setting up and maintaining the operational performance of a server and database.

Users: Persons who will use the Pericles system to run elections or vote. This includes Election Officers, Systems Administrators and Voters.

Voter: a person with the right to vote for a given election.

Voting Client: client software that allows authorized voters to cast and possibly change ballots by communicating with the Elections Server.

Weighted Vote: votes of unequal votes that are weighted based on criteria such as stake in a company.

XML: acronym for eXtensible Markup Language. It is a subset of the SGML document language designed for use on the Web and sanctioned by the W3C. It allows user defined tags.

11.2 References

MySQL - Paul DuBois - New Riders Publishing, copyright 2000.
http://avirubin.com/evoting.security.html
http://internetdollar.com/elections/elections.html
http://notablesoftware.com/evote.html
http://www.but.auc.dk/jdk-1.2b3/guide/rmi/SSLInfo.html
http://www.cert.org/tech_tips/denial_of_service.html
http://www.eli.sdsu.edu/courses/spring98/cs696/notes/
http://www.eli.sdsu.edu/courses/spring98/cs696/notes/rmiSecurity/rmiSecurity.html
http://www.javaworld.com/javaworld/jw-09-1999/jw-09-xmlmessage.html
http://www.mysql.com/doc/
http://www.netvoting.org/resources.htm
http://www.research.att.com/~lorrie/voting/hotlist.html

APPENDIX I—GUI INTERFACES

General note: all dialogue boxes for all of the below programs will have a "Help" button. This will bring up a help window that will explain the particular dialogue box that they are using. Each help window will have an "OK" button to clear the box.

AI.1) Elections Administrator

This program will be used by the system administrator to authorize the creation of elections, to suspend or restore elections, to enter systems limitations for elections and to validate elections officers. A series of screens will appear upon running the Elections Administrator:

1) The Elections Administrator Login Dialogue Box will ask the administrator to enter his/her password into a text field. The password information will be starred out as it is entered so that an observer will be unable to read it. There will be a button labeled "OK" to accept the information and another labeled "Quit" to quit the Elections Administrator. A message will appear if invalid identification information was entered, informing the user of the need to reenter the information. The Elections Administrator will automatically quit if invalid information is entered three times. This will be done in order to hinder automated attempts to break security.
2) The Elections Administrator Function Selection Dialogue Box will appear once valid identification is entered. There will be a choice of radio buttons and an "OK" button to accept the choice and a "Quit" button to quit the program. The six buttons will be as follows:
    a) Authorize New Election: This option will be used to authorize the creation of a new election. The Elections Officer(s) for the election being created must be present at this stage to enter their identification information. The Elections Administrator Authorize New Election Dialogue Box will appear when this option is selected. There will be several text fields that will be filled in by the Elections Officer. There will also be an "OK" button to accept the information and a "Cancel" button that will return the user to the Elections Administrator Function Selection Dialogue Box. The text fields will be as follows:
        i) Election Code: this will be the unique code used to identify the new election.
        ii) Number of Elections Officers: the number of elections officers authorized to run the election in question.
        iii) Minimum Elections Officers Needed: the minimum number of elections officers that are needed to enter identification information to make changes to the system.

    After the "OK" button is pressed, a new dialogue box will appear with the following text fields:
        1) Elections Officer User Name: this will be the unique name used by the system to identify the elections officer for the new election.
        2) Elections Officer Password: this will be the password used by the elections officer. The password information will be starred out as it is entered so that an observer will be unable to read it.
        3) Re-Enter Elections Officer Password: the elections officer must re-enter his or her password to verify that it was entered correctly. The password information will be starred out as it is entered so that an observer will be unable to read it.

The user will be informed what went wrong and asked to re-enter the information if the two passwords that are entered do not match or if the election code already exists on the system. This dialogue box will have an "OK" button. The dialogue box will appear once for each elections officer.

b) Restore Election: This option will be used to restore elections that were interrupted by a technical failure, a breach of security or an election suspension. The Elections Administrator Restore Election Dialogue Box will appear once for each of the minimum number of elections officers needed to effect changes and will ask for the election code and the login information of the elections officers. An e-mail will automatically be sent out to all voters registered for the election in question to inform them that service was interrupted and that they need to revote if they did not receive a verification number.

c) Suspend Election: This option will be used to suspend an election that has not begun yet or that is in progress. It will be used only in exceptional circumstances, such as an irretrievable breech of security. The Elections Administrator End Election dialogue box will appear once for each of the minimum number of elections officers needed to effect changes when this option is selected. There will be three text fields in this dialogue box, as well as an "OK" button to accept the information and a "Cancel" button that will return the user to the Elections Administrator Function Selection Dialogue Box. The three text fields will be:
    i) Election Code
    ii) Elections Officer User Name
    iii) Elections Officer Password: The password information will be starred out as it is entered so that an observer will be unable to read it.

    The user will be asked to re-enter the information if any of it is incorrect. If the information is correct, a warning dialogue box will appear telling the user that he or she is about to suspend an election. There will be a button saying "OK" to go ahead and another button marked "Cancel" that will return the user to the Elections Administrator Function Selection Dialogue Box. If an election is suspended, an e-mail will automatically be sent out to all voters registered in that election informing them that the election has been suspended.

d) Set Election Limitations: This option will be used to put global limitations on all elections that are created. The system administrator will be able to limit how long an election can last, how long data can be stored after elections end and the maximum size of enumeration lists. This will be used to prevent individual elections from tying up all of the resources of the system to the detriment of any other elections that are being held. Note that any limitations will only apply to elections that are registered after the limitations are entered and not to existing elections.

e) Change Administrator Password: This option will allow the system administrator to change his or her password by bringing up the Elections Administrator Change Administrator Password Dialogue Box. There will be three text fields and an "OK" button to accept the information and a "Cancel" button that will return the user to the Elections Administrator Function Selection Dialogue Box. The password information that is entered will be starred out as it is entered so that an observer will be unable to read it The three text fields will be:
    i) Old Password: this will be the system administrator's old password.
    ii) New Password: this will be the system administrator's new password.
    iii) Re-Enter Elections New Password: the user must re-enter his or her password to verify that it was entered correctly.

The user will be informed what went wrong and asked to re-enter the information if the two new passwords that are entered do not match or if the old password is incorrect.

AI.2) Election Editor

This program will be used by the elections officers to design ballots, set the settings for each election and enter enumeration lists. It will allow elections officers to change these settings once voters have been notified of their voting information. A series of windows will appear after running the Election Editor:

1) The Election Editor Login Dialogue Box will appear upon running the program and will ask the elections officer to enter information into three text fields. There will be a button labeled "OK" to accept the information and another labeled "Quit" to quit the Elections Editor. The text fields will be as follows:
    a) Election Code
    b) Election Officer User Name
    c) Election Officer Password: The password information will be starred out as it is entered so that an observer will be unable to read it.

    A message will appear informing the user of the need to reenter the information if invalid identification information was entered. The Elections Editor will automatically quit if invalid information is entered three times. This will be done in order to hinder automated attempts to break security. This login box will appear once for each election officer registered in that election, up to the minimum needed to effect changes for that election.
2) The Election Editor Function Selection Dialogue Box will appear once valid identification is entered. There will be a choice of several radio buttons and an "OK" button to accept the choice and a "Quit" button to quit the program. The radio buttons will be as follows:
    a) Edit Election Settings: This option will be used to set the preferences for an election once it is created. It will also be used to design the ballot for the election. An error will be given to the user if he or she attempts to alter these settings once voters have been notified of their voting information. The user will be asked to reenter any information if he or she exceeds any of the restrictions placed on elections by the system administrator. A succession of dialogue boxes will appear when this option is selected. Each will have a "Back" button and a "Next" button to move back and forth between the dialogue boxes to change information. The last dialogue box will have a "Done" button instead of a "Next" button. A list of the settings that were chosen and a sample of the ballot as it will appear to voters using the Voting Client will appear when the "Done" button is pressed. A button labeled "Accept" and another labeled "Cancel" will be present. Both of these buttons will return the user to the Election Editor Function Selection Dialogue Box, but the "Cancel" button will erase the information that was entered and the "Accept" button will store the information on the database. The dialogue boxes and the order in which they will appear are as follows:
        i) Select Election Dates Dialogue Box: This dialogue box will have a series of text fields as follows:
            1) Date to Publish Voter Information: The date and time that voters will be informed of the information that they need to vote.
            2) Date that Voting Begins: The date and time that voters may begin to vote.
            3) Date that Voting Ends: The last date and time that voters may vote.

4) Date to Publish Results: The date and time that the results of the election will be released.
5) Date to Remove Voter Information: The date and time that the enumeration list and all of the information relating to individual votes is deleted.

Each of these dates will be entered in the form YYYY-MM-DD hh:mm:ss. The user will be asked to reenter the dates and told why if the dates do not follow each other chronologically (e.g. the Date that Voting Ends is before the Date that Voting begins) when the "Next" button is pressed. There will also be two additional items on this dialogue box:

1) Time Window: a text box specifying the amount of time leeway that voters will be given to enter their votes because of differences in system times on different computers.
2) Time Zone: This will be a drop-down menu allowing the user to choose what time zone all of these dates correspond to.

ii) Select Election Options Dialogue Box: This dialogue box will present the user with a list of options, each with a check box. The default for all of these boxes will be unchecked. The options are as follows:

1) Allow voters to change their votes: This option will allow voters to change their votes if they have already voted.
2) Allow weighted votes: This option will allow elections to be held in which different voters have differently weighted votes.
3) Username specified: This option will allow usernames to be specified in the enumeration list rather than randomly generated, as is the default. This is to accommodate situations such as surveys where voting does not need to be done anonymously or where it is desirable to use existing information such as a valid student number as identification information.
4) Password specified: This option will allow usernames to be specified in the enumeration list rather than randomly generated, as is the default. This is to accommodate situations where it is desirable to use passwords that already exist on other systems.
5) Do not require password: This option allows for elections where voters do not need passwords, such as surveys.
6) Do not distribute ID information by e-mail: This will disable the default of sending election codes, user names and passwords out over e-mail to voters.
7) Allow refusal of ballots: This option allows voters to refuse their ballots before reading them.
8) Allow spoiled ballots: This option gives voters the option of spoiling their ballots.

There will also be several text fields to fill in:

1) Number of questions on ballot
2) E-mail address of elections contact: The e-mail address that will be distributed to voters so that they will have someone to contact with any questions that they might have. This will usually be the elections officer's e-mail address.
3) Comments: A place to enter any comments that will appear at the top of all ballots.

iii) Design Ballot Dialogue Box: This dialogue box will have four text fields:

1) Comments: Any comments preceding a question.

2) Question: A question that voters will be asked to vote on.
3) Number of Choices: The number of choices that correspond to this question.
4) Choices: The options that correspond to the question. Each new option must be followed by pressing the enter key.

The user will be asked to reenter the information if the number in the Number of Choices field does not correspond to the number of choices that were entered. This dialogue box will appear once for each question, up to the number of questions specified in the Select Election Options Dialogue Box.

iv) Select Election Statistics Dialogue Box: This dialogue box will allow the user to make choices about how the results of the election are posted. Each question will be displayed with the following options as checkboxes for it:
1) Report the Text of the Winning Choice
2) Report the Votes for each Choice
3) Report the Percentage Share for each Choice

The elections officer can thus choose to report all, some or none of these for each question on the general release web page that is produced. It should be noted that a web page containing all of this information will also be produced. There will also be several text boxes in this dialogue box:
1) E-mail address to send results files: leaving this blank means that the data gets e-mailed to all elections officers
2) Results site: Where voters should look to find the results of the election

v) Edit Enumeration List Dialogue Box: This dialogue box will allow the Elections Officer to enter the information for all registered voters. There will be several buttons on this dialogue box:
1) Load Enumeration List: This will allow the user to load a properly formatted enumeration list from disk. The user will be provided with a dialogue box to browse the disk. The information contained on the file will be appended onto any existing voters. The user will be asked to choose a new file if an improperly formatted file is given.
2) Clear Voter Data: This will erase any voter data that has already been entered
3) The "Next" and "Back" buttons.

vi) Preview Ballot Dialogue Box: This box will display the ballot exactly as it will appear to voters using the Voting Client

vii) Preview E-mail Dialogue Box: This dialogue box will show the format of the e-mail that will be sent to voters to inform them of their information, if this option is selected for the election in question.

b) Change Elections Officer Password: This option will allow the elections officers to change their passwords by bringing up the Elections Editor Change Elections Officer Password Dialogue Box. There will be three text fields and an "OK" button to accept the information and a "Cancel" button that will return the user to the Election Editor Function Selection Dialogue Box. The password information will be starred out as it is entered so that an observer will be unable to read it. The three text fields will be:
iv) Old Password
v) New Password

vi) Re-Enter Elections New Password: the user must re-enter his or her password to verify that it was entered correctly.

The user will be informed what went wrong and asked to re-enter the information if the two new passwords that are entered do not match or if the old password is incorrect.

AI.3) Elections Generator

This program will not have a GUI.

AI.4) Voting Client

This program will be used by voters to register and, if allowed in a particular election, change their votes. A series of windows will appear after running the Election Generator:

1) Voting Client Login Dialogue Box: The voter will be asked to enter some information in text fields upon running the program. There will be a button labeled "OK" to accept the information and another labeled "Quit" to quit the Voting Client. The text fields will be as follows:
    a) Server Address
    b) Election Code
    d) User Name
    e) Password: The password information will be starred out as it is entered so that an observer will be unable to read it.

   A message will appear informing the user of the need to reenter the information if invalid identification information was entered or if the client was unable to connect to the server. The Elections Editor will automatically quit if invalid information is entered three times. This will be done in order to hinder automated attempts to break security. The voter will receive a message denying him or her access if the server has a vote already registered from that voter and voters are not allowed to change their votes in this particular election.
2) Refuse Ballot Dialogue Box: Once the voter has successfully logged in, he or she will be presented with a dialogue box asking if he or she wishes to refuse the ballot as a form of protest. There will be one button marked "Yes" and another marked "No." This dialogue box will only appear if refusing the ballot has been allowed in the particular election
3) Voting Dialogue Box: The client will then display the ballot to the client with a set of radio buttons for each choice. The option of spoiling a question will be given for each question if this has been permitted in the settings for the particular election. There will be no default selection for any of the choices. There will be a button at the bottom of the ballot labeled "Submit" that will submit the ballot to the server and another labeled "Quit" which quits the program.
4) Vote In Transit Display Box: After pressing the "Submit" button, the voter will be presented with a display box telling him or her that the vote has been submitted. The voter will be asked to wait for confirmation that the vote has been submitted. The client will time out if confirmation has not been received in three minutes (this is more than enough time if everything is going well, given reasonable network traffic). If this is the case, a dialogue box will appear informing the voter that his or her vote was not registered, and asking him or her to try again.
5) Vote Accepted Dialogue Box: Once the server has accepted and recorded the vote, a message will be displayed to the client saying that the vote was successfully recorded along with a verification number provided by the server. There will be two buttons, one

labeled “Vote Again”, which will bring up the Voting Client Login Dialogue Box and another labeled “Quit” which will quit the program.

APPENDIX II—DATABASE TABLES

This appendix outlines the tables used by the relational database. Each title corresponds to a table name and the items under each title are the names of the columns of the table. The rows stored in each table will be the information for all elections using the database. It might be advisable to set a future version of this software to use different databases for each election if the elections are on a very large scale.

Officers: contains the identification information for all elections officers.

1) ElectionCode: the code identifying the elections that the election officer is associated with
2) OfficerName: the username of the elections officer
3) OfficerPassword: the password of the elections officer
4) ElOfEMail: the e-mail address of the elections officer

Voters: contains the identification information for all voters

1) EmailID: a code used to link to the Emails table in case it is desirable to be able to send messages to individual voters. This will normally be set to 0 for all rows, in order to make it impossible to associate votes to voters.
2) VoterName: the username of the voter (randomly generated or specified)
3) VoterPassword: the password of the voter (randomly generated or specified)
4) ElectionCode: the code identifying the elections that the voter is associated with
5) VerificationNum: a code generated by Elections Server Objects that can be used to verify that a voter voted. Can potentially be used by voters to verify how they voted.
6) ResultID: a code used to link to the VoteAnswers table.
7) Weighting: how many votes the voter's vote counts for. The default is one. In elections where voters are not permitted to change their votes after having voted, this value will be set to 0 as soon as a voter has voted and his or her vote is recorded.

EMails: contains the information necessary to send e-mails to voters

1) EmailID: a code used to link to the Voters table in case it is desirable to be able to send messages to individual voters. This will normally be set to 0 for all rows, in order to make it impossible to associate votes to voters.
2) ElectionCode: the code identifying the elections that the voter is associated with
3) Email: the e-mail address of the voter

ElectionsAdmin: contains the information needed by the Elections Generator to administrate elections

1) ElectionCode: the code identifying the elections that the information is associated with.
2) DateOfInfoRemoval: the date and time the voter information will be erased from the database
3) DateToSendEmails: the date and time on which e-mails will be sent to voters informing them of the election and their identification information. Set to NULL if no e-mails are to be sent.
4) VoteBeginDate: the date and time on which votes will begin to be accepted for an election
5) VoteEndDate: the date and time on which votes will no longer be accepted for an election

6) TimeWindow: the amount of time leeway that will be allowed to compensate for diffences in systems clocks
7) TimeZone: the time zone corresponding to all dates and times given out by the elections software
8) EmailOfElectionContact: the e-mail address of the person who voters can contact
9) Suspend: a value specifying whether or not the Elections Generator should suspend an election, should restore an election or should take no such action

Publishings: contains the information needed by the Elections Generator to release results
1) ElectionCode: the code identifying the elections that the information is associated with.
2) PublishEMail: where to e-mail the results. The default will be NULL, which means that the results will be sent to all of the elections officers.
3) PublishDate: the date and time on which the results statistics will be released.
4) ResultSite: the web site that voters will be told to look for statistics.

VoteAnswers: the votes registered by voters
1) ResultID: a code used to link to the Voters table
2) Weighting: how many votes the voter's vote counts for.
3) QuestionID: a code used to identify the particular question being voted on
4) Answer: the vote of the voter. SP will be used for spoiled and RE for refused.

Statistics: the statistics that need to be released when the results are published
1) ElectionCode: the code identifying the elections that the statistics are associated with.
2) DisplayWinner: whether or not to display the winning choice of the questions. 1 for yes and 0 for no.
3) DisplayNumbers: whether or not to display the number of votes for each choice. 1 for yes and 0 for no.
4) DisplayPercentages: whether or not to display the percentages of votes for each choice. 1 for yes and 0 for no.

ServerStats: statistics relating to network traffic. This may or may not be implemented.
1) ElectionCode: the code identifying the elections that the statistics are associated with.
2) NumberOfHits: the total number of hits registered so far by the Elections Server Object for the election in question.

APPENDIX III—BALLOT FORMATS

This appendix outlines the XML formats that will make up the messages exchanged between the Voting Client and the Election Server Objects.

AIII.1) Login Rejected Ballot

This ballot will be sent from the Election Server to the Voting Client if a login attempt fails:

```
<ballot>
        <login_fail>
        You were unable to login to the Elections Server. You did successfully contact the
        elections server, but the login attempt failed because
                <fail_reason>
                        <!-- possible reasons could be: invalid election code, incorrect
                        password or username, server too busy, voter has already voted,
                        server down, election suspended -->
                </fail_reason>
        Please try reentering the correct information, or contact
                <contact> electionguy@usgovernment.com </contact>
        </login_fail>
</ballot>
```

AIII.2) Empty Voting Ballot

This ballot will be sent from the Election Server to the Voting Client if a login attempt succeeds:

```
<ballot>
        <header>
                <contact> electionguy@usgovernment.com </contact>
                <refusal> yes </refusal>
                <spoil> yes </yes>

                <!-- A comment to appear at top of ballot -->
                <topcomment>
                        This ballot is secure and all information transmitted
                        confidential.  The US government knows what it is doing
                        and you have nothing to fear.
                </topcomment>
        </header>

        <question>
                <!-- an optional comment can be added to each question -->
                <opt_text>
                        This question will determine your next president
                        of the United States.
                </opt_text>
                Who do you wish to vote for to be President?
```

```
            <choice> a) Jim Bob </choice>
            <choice> b) Joe Smith </choice>
            <choice> c) Betty Sue </choice>
            <choice> b) Joe Smith </choice>
        </question>

        <pgbr/> <-- put next question on another page(screen) -->

        <question>
            <!-- etc. for each question on the ballot -->
        </question>
</ballot>
```

AIII.3) Filled in Voting Ballot

This ballot will be sent from the Voting Client to the Elections Server after a successfully received voting ballot has been filled in by a voter:

```
<ballot>
        <header>
            <contact> electionguy@usgovernment.com </contact>
            <refused> no </refused>
            <electcode> elec359 </electcode>
            <useridd> rjones </userid>
            <psswd> 43kl33n9 </psswd>

            <!-- A comment to appear at top of ballot -->
            <disclamer>
                This ballot is secure and all information transmitted
                confidential.  The US government knows what it is doing
                and you have nothing to fear.
            </disclamer>
        </header>

        <question>
            <!-- an optional comment can be added to each question -->
            <opt_text>
                This question will determine your next president
                 of the United States.
             </opt_text>

             Who do you wish to vote for to be President?
            <!-- "chosen" indicates which choice is selected and a spoiled ballot is
            indicated by no chosen tag at all -->
            <choice> a) Jim Bob </choice>
            <chosen> b) Joe Smith </chosen>
            <choice> c) Betty Sue </choice>
            <choice> b) Joe Smith </choice>
        </question>

        <pgbr/> <-- put next question on another page(screen) -->
```

```
        <question>
        <!-- etc. for each question on the ballot -->
         </question>
</ballot>
```

AIII.4) Vote Could not be Registered Ballot

The following ballot will be sent from the Elections Server to the Voting Client when a filled in ballot cannot be successfully recorded:

```
<ballot>
        <vote_fail>
        Your vote could not be registered.  Please try again or contact
                <contact> electionguy@usgovernment.com </contact>
        The problem is
                <fail_reason>
                        <!-- possible reasons could be: server too busy, voter has already voted, server down, election suspended -->
                </fail_reason>
        </vote_fail>
</ballot>
```

AIII.5) Vote Successfully Registered Ballot

The following ballot will be sent from the Elections Server to the Voting Client when a vote has been successfully recorded on the database:

```
<ballot>
        <vote_succ>
                Your vote was recorded successfully. Your reference number is
                        <ref_num> 48759321 </ref_num>
                Thank you for using the Pericles Election Software
         </vote_succ>
</ballot>
```

APPENDIX IV—E-MAIL FORMATS

This appendix shows what the various e-mails sent out by the system will look like.

AIV.1) Voter Registered E-mail

This e-mail will be sent out automatically to inform voters that they are registered to vote in an election and will tell them their login information. This option may be disabled by elections officials. The format will be as follows:

> To: <voter e-mail address>
> From: <contact e-mail address>
> Subject: electronic election registration
>
> You have been registered to vote in an electronic election using the Pericles Voting Client. Please contact <contact e-mail address> if you have any question about the election, if you have difficulty voting or if you need access to a Pericles Voting Client.
>
> Enter the following information after running the Pericles Voting Client:
>
> Server Address: <IP address of the Server Object>
> Election Code: <election code of the election>
> User Name: <user name of the voter>
> Password: <password of the voter>
>
> It is essential that you do not share any of this information with anyone else, as they will then be able to vote for you! Your vote will not be officially counted unless the Pericles Voting Client gives you a verification number after voting. Be sure that you do not leave your computer between the time that you log in to vote with the Voting Client ant the time that you receive a verification number, as someone could use your computer to vote in your place while you are gone. Note that some elections do not require a user name or password. If you have not been given a user name and password in this election, then simply leave these boxes blank when you use the Voting Client.
>
> You may only vote between the following dates, which are in the <time zone of server> time zone: <election start date> and <election end date>. You <may / may not> change your vote after sending it. Your vote counts as <weighting of vote> vote(s).
>
> The results will be posted on <posting of results date> at <URL of results>.
>
> Thank you for using the Pericles Elections Software.

AIV.2) Election Halted E-mail

This e-mail will be sent if an election is halted by an elections officer or if the Elections Generator notices that an Elections Server Object is no longer operating as it should. The format will be as follows:

> To: <voter e-mail address>

From: <contact e-mail address>
Subject: electronic election registration

The voting for election <election code of the election> has been halted and no more votes may be registered at this time. Please contact your local election officials for more information. Another e-mail will be sent if and when the election starts again.

We apologize for any inconvenience.

AIV.2) Election Restarted E-mail

This e-mail will be sent if an election is started again after having been interrupted. The format will be as follows:

To: <voter e-mail address>
From: <contact e-mail address>
Subject: electronic election registration

The voting for election <election code of the election> has been restored. If you have received a verification number for your vote, then your vote has been properly recorded. If not, please try voting again

We apologize for any inconvenience.

APPENDIX V—Enumeration List Format

The enumeration list is a file containing authorized voters that must be entered by the elections officer prior to any election. It is a comma delimited list. Each line corresponds to a voter. There are four fields, which must appear in the following order:

e-mail address, vote weighting, pre-defined username, pre-defined password

All of these fields are optional, and some or all may be left blank. However, the three commas must always be present.

APPENDIX VI—Log File Formats

It is important to have log files in order to recover from failures of the system without loss of data. The log files that will be kept for the Pericles Elections Software will be used to recover from failures or crashes of the Election Server that occur before data is recorded in the database. These log files will not account for failures of the database itself, as to do so would compromise the security and privacy of the data. It is assumed that the database itself provides a method to recover data after a database crash.

Only the systems administrator will have access to the log files and only he or she will have the ability to delete it when it is deemed fit. The log files will be simple text files that are semi-colon delimmitted. There will be two types of log files:

AVI.1) Elections Editor Log File

This file, named elections.log, will provide a record of all changes that are made by elections officers. This is important because it provides a method of tracing changes that are made and who made them in case there are breaches of elections editor security.

This log file will have the following format:

ChangeTime; IPAddress; Election Code; OfficerName; Behaviour; State

a) ChangeTime = the date and time that the change was made (YYYY-MM-DD hh:mm:ss)
b) IPAddress = the IPAddress of the machine that was used to make the change
c) OfficerName = the user name of the elections officer that is making the change. If there are multiple elections officers that need to log in to make changes, this will be the first one.
d) Behavior = the action taken by the elections officer. The options are:
    1) login = login to the system
    2) suspend = suspend an election
    3) restore = restore an election
    4) create = define the settings for a new election
    5) edit = edit the settings for an existing election
e) State = how far the elections officer got
    1) sent = data was sent to the database or the Elections Server
    2) recorded = data was successfully recorded on the database or the Elections
    3) success = successful login (for logins only)
    4) fail = unsuccessful long (for logins only

The following is a sample section of an elections.log log file:

2001-05-20 20:58:12; 130.23.17.5; 534; Pelopenesian; login; fail
2001-05-20 20:58:56; 130.23.17.5; 534; Peloponnesian; login; success
2001-05-20 20:59:32; 130.23.17.5; 534; Athens; login; success
2001-05-20 20:59:49; 130.23.17.5; 534; Sparta; login; success
2001-05-20 21:21:12; 130.23.17.5; 534; Peloponnesian; create; sent
2001-05-20 21:21:12; 130.23.17.5; 534; Peloponnesian; create; recorded

AVI.1) Voting Log File

This file, named voting.log, will provide a record of all votes that are made. It will include the e-mail addresses of voters, in case they need to be contacted to revote, but it will not include the details of their vote. This is to prevent someone from telling how an individual voted based on the log file. The IP Address of the voter will also be recorded in order to help locate voters that are trying to breach the security of the system or make denial of service attacks.

This log file will have the following format:

ChangeTime; IPAddress; ElectionCode; Email; Behavior; State

a) ChangeTime = the date and time that the vote was made (YYYY-MM-DD hh:mm:ss)
b) IPAddress = the IPAddress of the machine that was used to vote
c) Email = the e-mail address of the voter.
d) Behavior = the action taken by the voter. The options are:
    i. login = attempt by the voter to login
    ii. emptyballot = an emptyballot was sent to the voter
    iii. completedballot = a completed ballot was returned to the Elecitons Server
    iv. votestored = the vote was stored on the database
    v. verificationsent = a verification number was sent to the voter
e) State = how far the elections officer got
    i. success = the action was successful
    ii. failure = the action failed

The following is a sample section of a voting.log log file:

2001-06-15 23:58:22; 130.23.17.5; 534; Socrates; login; success
2001-06-15 23:58:45; 130.23.17.5; 534; Socrates; emptyballot; success
2001-06-15 23:59:11; 130.27.57.5; 412; Plato; login; success
2001-06-15 23:59:16; 130.27.57.5; 412; Plato; sendballot; success
2001-06-16 00:05:09; 130.23.17.5; 534; Socrates; completedballot; success
2001-06-16 00:05:12; 130.23.17.5; 534; Socrates; votestored; success
2001-06-16 00:06:12; 130.27.57.5; 412; Plato; completedballot; success
2001-06-16 00:06:17; 130.23.17.5; 534; Socrates; verificationsent; success
2001-06-16 00:06:22; 130.27.57.5; 412; Plato; votestored; success
2001-06-16 00:06:24; 130.27.57.5; 412; Plato; verificationsent; success

APPENDIX VII—Results File Formats

There will be two results file produced for each election—one a complete list of elections statistics and the other a list of statistics showing only the results selected for reporting by the elections officer(s). Both files will be in the HTML format and will be automatically e-mailed to the specified address (the default is all elections officers). The following is an outline of a complete results file. A limited results file could be missing the lines indicating the text of the winning choice, the total number of votes for each choice and/or the percentage of votes for each choice.

```
<HTML>
<HEAD>
        <META NAME="Author" CONTENT="Pericles Elections Server">
        <TITLE>results</TITLE>
</HEAD>
<BODY>

<H1>Complete Results for Election &lt;ELECTION CODE></H1>
This election ran from &lt;START DATE> to &lt;END DATE> on the Pericles
Election Server. These results were released on &lt;STATS RELEASE DATE>.
<P>&lt;NUMBER OF BALLOTS REFUSED> ballots were refused.<BR>
<HR WIDTH="100%">

<H3>The results for Question 1:</H3>
<BLOCKQUOTE>Question: &lt;TEXT OF QUESTION>
<BR>Winner: &lt;TEXT OF WINNING CHOICE>
<BR>Spoiled votes: &lt;NUMBER OF SPOILED VOTES> (&lt;% OF SPOILED VOTES>)
<P>CHOICE A: &lt;TEXT OF CHOICE A>
<BLOCKQUOTE>&lt;NUMBER OF VOTES FOR A> votes were cast for this choice.
<BR>This represents &lt;PERCENTAGE OF VOTE>% of the votes.
<BR> </BLOCKQUOTE>
CHOICE B: &lt;TEXT OF CHOICE A></BLOCKQUOTE>

…

<HR WIDTH="100%">
<H3> The results for Question 2:</H3>

…

</BODY>
</HTML>
```

APPENDIX VIII—Configuration File Format

The following is an example of the format of the configuration file used by the Elections Server. It is the responsibility of the Systems Administrator to fill it out when installing the Pericles system. Instructions on how to do this will be provided in the install readme file.

<config>

<dbname> Election_Data </dbname>
<dbhost> media21.cis.uoguelph.ca </dbhost>
<dblogin> My_Database <\dblogin>
<dbpass> storestuff <\dbpass>
<dbport> 99 </ddbport>

<salogin> admin </salogin>
<sapassword> r8w735 </sapassword>
<saemail> admin@pericles.com </saemail>

<votelogloc> pericles/log_files </votelogloc>
<eleclogloc> pericles/log_files </eleclogloc>

<resultloc> pericles/results_files </resultloc>

</config>