```
name: Team Dijkstra
date: April 16, 2013
program: Email with ACL2
instructor: Dr. Rex Page
language: ACL2, C++, Shell Scripting, XML

actual added lines: 401
actual base lines: 66
actual modified lines: 15
actual removed lines: 0

new objects:

        - name: xmlScanner
          estimated lines: 18
          type: NonIO

        - name: xmlParser
          estimated lines: 18
          type: NonIO

        - name: getSubject
          estimated lines: 9
          type: NonIO

        - name: getFrom
          estimated lines: 9
          type: NonIO

        - name: getTo
          estimated lines: 9
          type: NonIO

        - name: getEmail
          estimated lines: 14
          type: IO

        - name: getInboxFiles
          estimated lines: 18
          type: NonIO

        - name: removeInboxFile
          estimated lines: 18
          type: NonIO

        - name: registerClient
          estimated lines: 18
          type: NonIO

        - name: writeEmail
          estimated lines: 18
          type: NonIO

        - name: writePreferences
          estimated lines: 18
          type: NonIO

        - name: getBlocked
          estimated lines: 14
          type: IO

        - name: getTag
          estimated lines: 14
```

```
    type: IO

  - name: parseMessage
    estimated lines: 18
    type: NonIO

  - name: parseSpamKey
    estimated lines: 18
    type: NonIO

  - name: toServer
    estimated lines: 9
    type: IO

  - name: fromServer
    estimated lines: 9
    type: IO

  - name: getErrors
    estimated lines: 18
    type: NonIO

  - name: generateSpamKey
    estimated lines: 18
    type: NonIO

  - name: generateMessage
    estimated lines: 18
    type: NonIO

  - name: generateRegRequest
    estimated lines: 18
    type: NonIO

  - name: getMessageLength
    estimated lines: 18
    type: NonIO

  - name: writeHeaderInfo
    estimated lines: 18
    type: NonIO

  - name: parseHeaderInfo
    estimated lines: 18
    type: NonIO

  - name: getContentType
    estimated lines: 9
    type: NonIO

  - name: setContentType
    estimated lines: 9
    type: NonIO

  - name: ReadMsgInput
    estimated lines: 16
    type: IO

  - name: readRegInput
    estimated lines: 16
    type: IO

  - name: writeToAddressBook
    estimated lines: 16
    type: IO

  - name: writeToClientInbox
    estimated lines: 16
    type: IO

  - name: processMessage
    estimated lines: 18
    type: NonIO

  - name: processRequest
    estimated lines: 18
    type: NonIO

  - name: isSpam
    estimated lines: 9
    type: NonIO

  - name: isBlockedUser
    estimated lines: 6
    type: NonIO

  - name: parseAddress
    estimated lines: 6
    type: NonIO

  - name: createUserGroup
    estimated lines: 14
    type: IO

  - name: createMailingList
    estimated lines: 16
    type: IO

  - name: updateMailingList
    estimated lines: 16
    type: IO

  - name: getAvailableMailingList
    estimated lines: 16
    type: IO

  - name: getMailingListRequest
    estimated lines: 9
    type: IO

  - name: getMailingListRegistrationRequest
    estimated lines: 16
    type: IO

  - name: isInUserGroup
    estimated lines: 18
    type: NonIO

  - name: isInMailingList
    estimated lines: 18
    type: NonIO

  - name: isRegistered
    estimated lines: 18
    type: NonIO
```

```
- name: isNameAvailable
  estimated lines: 18
  type: NonIO

- name: getMessageLength
  estimated lines: 9
  type: NonIO

- name: readInbox
  estimated lines: 16
  type: IO

- name: storeClientName
  estimated lines: 14
  type: IO


- name: getAddress
  estimated lines: 9
  type: Non IO

- name: email
  estimated lines: 14
  type: IO

- name: inbox
  estimated lines: 16
  type: IO

- name: register
  estimated lines: 16
  type: IO

- name: blockUser
  estimated lines: 18
  type: NonIO

- name: tagMessage
  estimated lines: 18
  type: NonIO

- name: getServerAddressBook
  estimated lines: 16
  type: IO

- name: getLocalAddressBook
  estimated lines: 18
  type: NonIO

- name: updateAddressBook
  estimated lines: 18
  type: NonIO

- name: getMailingLists
  estimated lines: 16
  type: IO

- name: createMailingList
  estimated lines: 16
  type: IO

- name: registerForMailingList
  estimated lines: 16
  type: IO

- name: isRegistered
  estimated lines: 18
  type: NonIO

- name: isNameAvailiable
  estimated lines: 18
  type: NonIO

- name: isInAddressBook
  estimated lines: 18
  type: NonIO

- name: addSpamFilter
  estimated lines: 9
  type: NonIO

- name: addSpamKeyword
  estimated lines: 9
  type: NonIO

- name: addSpamAddress
  estimated lines: 9
  type: NonIO

- name: getDomain
  estimated lines: 6
  type: Non IO

- name: getName
  estimated lines: 6
  type: Non IO

- name: getPassword
  estimated lines: 6
  type: Non IO

- name: registerUser
  estimated lines: 14
  type: IO

- name: getEmailXML
  estimated lines: 18
  type: NonIO

- name: getEmailStructure
  estimated lines: 9
  type: NonIO

- name: getEmailStructureList
  estimated lines: 9
  type: NonIO

-       name: consume
        estimated lines: 6
        type: Non IO

-       name: tag
        estimated lines: 6
        type: Non IO

-       name: pcData
```

```
                estimated lines: 4
                type: Non IO

        -       name: nextToken
                estimated lines: 9
                type: Non IO

        -       name: tokenizeXML
                estimated lines: 3
                type: Non IO

time log:

       - date: Jan 17, 2013
         start time: 10:30AM
         end time: 11:45AM
         phase: Conception
         comment: Team Dijkstra's regular meeting time. We brainstormed project
ideas and decided on a networked chat client and server system.

-       date: January 18, 2013
                start time: 1:00am
                end time: 2:00am
                phase: Testing
                comment: Tested network connectvity for ACL2 using various methods.
Found that through file streaming, we could transfer files through connected network
drives.  We are not able to connect to a specific TCP/IP address or anything without
a directory resolution on the network.  Example: \\MatthewCrist-Laptop would work,
\\127.0.0.1 would work, but http://127.0.0.1 or \\127.0.0.1:80 would not yeild
results.  I have determined that either we will need supplemental language in order
to make the transfer or map network drives.


       - date: Jan 22, 2013
         start time: 10:30AM
         end time: 11:45AM
         phase: Conception
         comment: Team Dijkstra's regular meeting time. Finalized the idea of the
chat system and began initial designs for the proposal.

        - date: Jan 22, 2013
          start time: 7:32PM
          end time: 8:44PM
          phase: Conception
          comment: Worked on updating the team's LoC table to include tImpl from
last semester. Reused the spreadsheet from last semester and added the new data.
Then included the information into the t2 and t3 documents.

       - date: Jan 22, 2013
         start time: 8:53PM
         end time: 10:49PM
         phase: Testing
         comment: Worked on researching networking from within ACL2. There is not
any native support for networking. However, we did discover that we could use
networking if the Operating System supports it. Since our operating systems can be
networked, we can work around this limitation.

-       date: January 23, 2013
                start time: 2:30am
                end time: 4:00am
                phase: Testing
                comment: Attempted to research means by using Common Lisp to
implement a layer for TCP/IP transfer to see if this is a viable alternative.  The
```

```
solutions that are available seem to be beyond the scope to which we can implement.
It appears the network drive route is our best alternative.


- date: Jan 23, 2013
          start time: 8:04PM
          end time: 10:27PM
          phase: Conception
          comment: Worked on writing the content of the Proposal. Finished the
sections for the Overview, High-level design, some of the requirements, and began
the PROBE estimate.

       - date: Jan 24, 2013
         start time: 10:30AM
         end time: 11:45AM
         phase: Conception
         comment: Team Dijkstra's regular meeting time. We worked on the proposal
and worked on several proofs of concepts to determine if the project is feasible.

        -       date: January 25, 2013
                start time: 10:00am
                end time: 10:20am
                phase: Testing
                comment: Researched event handling in ACL2 from previous project
regarding "Blue Ball" scenario.  Determined that this could be used to implement the
client side gestures for the chat messages to be sent and received.  Also noted that
the primitive nature of the interface would require that we construct many of the
GUI element ourselves.  I think we should be able to assign functions to these
events to read and write files where necessary.

        - date: Jan 26, 2013
          start time: 4:04PM
          end time: 6:31PM
          phase: Conception
          comment: Finished the complete proposal with all the specified
requirements and PROBE estimate.

       - date: Jan 29, 2013
         start time: 10:30AM
         end time: 11:45AM
         phase: Conception
         comment: Team Dijkstra's regular meeting time. Dr. Page addressed concerns
about the Project and that GUI's and File IO cannot exist in the same ACL2 program.
We began a major re-design of the project and re-wrote most of the project
requirements that are to be included in the proposal.

        - date: Jan 29, 2013
          start time: 11:45AM
          end time: 2:12PM
          phase: Conception
          comment: Found out that some of the components of the original proposal
were not feasible. Worked with Matthew to re-write sections of the proposal to
include the new ideas for the project and write out the requirements for the
project. The new project will be an email system instead of a chat system, with more
emphasis on message delivery and content.

        -       date: January 29, 2013
                start time: 12:00pm
                end time: 2:30pm
                phase: Conception
                comment: After discussion with Rex Page regarding some of the
features of ACL2, he described to us that ACL2 cannot use a GUI and write/read from
files at the same time.  This has thrown us in a tizzy about writing a chat client.
```

```
After reassessment of the situation, we determined that we could salvage much of the
information that we derived in previous sessions by using supplemental language,
such as C to write the server and client interfaces that would invoke ACL2
executables on demand and have these applications handle the events for both client
and server processes.  We could use folder monitoring on server side and invoke
those modules that are created adding to the modularity of the application itself.
Isaac, Wes and I rewrote the design document so that it would be available for
Thursday (January 31, 2013) for the presentation.  We also determined that Adam
should be the one to give the overview of the application since he may want the
extra speaking opportunity.

        -       date: January 30, 2013
                start time: 4:00am
                end time: 5:30am
                phase: Conception
                comment: Created the slides for the presentation that will be held
tomorrow (January 31, 2013).  Used the design document as a point of reference for
bullet points.  Also devided the slide information based on role (as opposed to
design layer) to "sell" the idea to the end user and convey the purpose of the
process we are intending to use.

        -       date: January 31, 2013
                start time: 8:00am
                end time: 9:30am
                phase: Testing
                comment: Attempted to get Proofpad to function on my windows PC
(Microsoft Windows 8) to no success.  Attempted to get it to work on my Linux
partition as well (Ubuntu 11.04 LTS) with no such luck either.  Both result in
NullPointerExceptions being tossed back by the application itself.  Unable to invoke
ACL2 internally as a result.  Decided to do some testing with ACL2 command line and
get familiar with the environment  Noticed that the teachpacks were not certified
for io-utilities.lisp and list-utilities.lisp, so I had to include them in a
subfolder and include them in the file directly.  Cannot use program mode in ACL2
and have to stay in logic mode else the files will not include correctly into the
project.  Considering DrRacket as the IDE for development as a result.

        -       date: January 31, 2013
                start time: 1:00pm
                end time: 2:30pm
                phase: Coding
                comment: Started to develop the infrastructure for the server
monitor program.  Determined that the use of processes and a dependency relationship
was necessary to ensure that a process was complete before invoking a dependent
processes.  Derived an XML DTD for the input format to load modules into the server
environment for invocation.  Assigned names to modules in the format
(program).(content).(action) naming convention, a folder to monitor for files, and a
process to invoke when a file is detected.  Information has been updated on the wiki
for reference on module creation under >> The Server Monitor.

        -       date: February 1, 2013
                start time: 4:30am
                end time: 5:20am
                phase: Coding
                comment: Did some more work on the Server Monitor.  Wrote threading
for directory monitoring and process invocation.  Have not tied module invocation
directly to program yet, as I need to verify that executables can be effectively
created from Racket.  File size seems to be rather large at the moment for just a
hand full of code.  May consider an alternative approach.

        - date: Feb 3, 2013
          start time: 12:42AM
          end time: 2:09AM
          phase: Conception
          comment: Matthew emailed about an issue with the ACL2 EXE's not working
correctly and possibly being unable once the project gets large. Spent this time
researching how to move the project to UNIX based servers and working with launching
ACL2 from the terminal and using shell scripts to call ACL2 lisp files.

        -       date: February 3, 2013
                start time: 10:00am
                end time: 12:30pm
                phase: Testing
                comment: Determined that executables in ACL2 is not a practical
solution.  Tried invocations through redirection on the input method for ACL2 since
the executable did not take any arguments.  So far I have had success.  Module
invocation can occur through a shell script which does not require dependency checks
(which will allow for linear execution.)  Reassessing the use of unsynchronized
process invocation.  Perhaps synchronizing the main processing thread to halt while
a process is running could be the best alternative.  Invocation of a shell script
also alleviates the overhead that may have been involved and we can implement some
of the operating system features to transfer information (such as ftp for file
transfer).

        - date: Feb 4, 2013
          start time: 5:23PM
          end time: 6:47PM
          phase: Conception
          comment: Worked on typing the progress report with the completed task to
date and writing out the plans for the task to be completed. Also noted in in the
progress report the need to move the system server to a UNIX based system to run the
server components

        -       date: February 5, 2013
                start time: 4:00am
                end time: 5:30am
                phase: Documentation
                comment: Updated progress report with findings over the last two
weeks for delivery to Dr. Page and the team.  Need to address issues with the change
in the group make up, since Isaac will be unable to participate in the project until
his personal issues are resolved.

        - date: Feb 5, 2013
          start time: 10:30PM
          end time: 11:45PM
          phase: Conception
          comment: Team Dijkstra's regular meeting time. We delivered a progress
report to Dr. Page. We worked on converting the Windows based designs to a UNIX file
system in order for the system to run.

        - date: Feb 5, 2013
          start time: 7:32PM
          end time: 8:35PM
          phase: Conception
          comment: Worked on setting up ACL2 modules to run though the UNIX shell
and wrote test scripts to automatically call ACL2 functions from a shell script

        -       date: February 6, 2013
                start time: 10:00am
                end time: 11:00am
                phase: Coding
                comment: Reorganized server code into better directory management so
ensure a structure that can be deployed more effectively.  Updated references to be
on local scope, as opposed to global folder references for portability purpose.

        -       date: Februaru 7, 2013
                start time: 6:30am
                end time: 8:00am
                phase: Coding
                comment: Wrote scanner portion for the analyzer for XML content that
can be parsed with ACL2.  Added consume, tag, pcData, nextToken, and tokenizeXML
functions.  Entry point is tokenizeXML which takes a string of XML and will return a
list of tokens and the token type.  This should then be added to a stack to verify
xml correctness and can then be used to extract the PCDATA information contained
between the brackets.

        - date: Feb 7, 2013
          start time: 10:30PM
          end time: 11:45PM
          phase: Conception
          comment: Team Dijkstra's regular meeting time. We worked on the XML format
and multi-level design and designed the data structure format for the project. We
then wrote sections of the progress report and assigned an XML module to each team
member.

        - date: Feb 9, 2013
          start time: 5:14PM
          end time: 7:32PM
          phase: Conception
          comment: Worked on designing the XML structure for email messages.
Designed the XML layout, Document type definition and explanation. These topics were
then uploaded to the groups wiki for review, comment, and inclusion in the design
document.

        - date: Feb 11, 2013
          start time: 5:12PM
          end time: 8:04PM
          phase: Conception
          comment: Worked on the team's design document and formatting of the
sections. Took the information from the group's wiki and generated the data
structure and IO format sections from this information. The other sections were
expanded from the initial proposal with updates to relevant sections where the
server information had been changed.

        - date: Feb 12, 2013
          start time: 10:30PM
          end time: 11:45PM
          phase: Conception
          comment: Team Dijkstra's regular meeting time. We reviewed the team design
document and made many changes to the XML format and found several errors. We marked
up the design for submission on Thursday.

        -       date: February 14, 2013
                start time: 12:15am
                end time: 12:30am
                phase: Conception
                comment: Derived the format, in XML, for the transportation of the
registration of a user into the address book for the server.  Also formulated the
steps to be involved in order to invoke the appropriate modules.

        -       date: February 14, 2013
                start time: 12:30am
                end time: 1:45am
                phase: Coding
                comment: Wrote the shell script that acquired the contents of the
source file (ACL2 definitions) and compiled a function invocation to be written to a
temporary file, input directed into ACL2 and removal of the temporary file created.
Considering just including the book to the source file and writing a new temporary
file that invokes the function and includes to appropriate books in order to not
manipulate the source files for unexpected results.
```

```
        -       date: February 14, 2013
                start time: 2:00am
                end time: 4:00am
                phase: Coding
                comment: Created functions for managing address-book.  getAddress,
parseAddresses, getAddressBook acquire values from the XML that was passed to the
script via redirection through the XML that was acquired through the shell
scripting.

        -       date: February 14, 2013
                start time: 6:15am
                end time: 8:00am
                phase: Coding
                comment:        Created the functions for managing output of the
address-book data structure.  addressXML, addressBookXML and getAddressBookXML will
acquire the XML for the structure to be written back to the xml data file.
Determined that I will need to write to a temporary file and delete the original
source, and rename the temporary file to the permanent persistent file in order to
allow for the file to be updated (since I cannot write back to a read file).
Updated shell script to reflect these changes in
./server/modules/user/register/register-user.sh.

        -       date: February 16, 2013
                start time: 1:15am
                end time: 3:00am
                phase: Coding
                comment: Added functions that acquires the contents of the tokens
passed by parsing the XML information.  getDomain, getName, getPassword all use
these tokens in order to extract the information from the registration file that
will be used to determine what information needs to be added to the address-book.

        -       date: February 16, 2013
                start time: 3:00am
                end time: 5:00am
                phase: Coding
                comment: Added functions that will test the address existence (to
prevent duplication) and add/remove an address from the address-book.
isInAddressBook tests the predicate conditions to determine if an address can be
added/removed by addAddress/removeAddress functions.

        - date: Feb 17, 2013
          start time: 3:43PM
          end time: 4:51PM
          phase: Coding
          comment: Worked on designing the server email components of the server
module. Wrote out data flow diagrams to match the data structure format and XML I/O
from the design document, and wrote function prototypes.

        - date: Feb 18, 2013
          start time: 5:05PM
          end time: 6:12PM
          phase: Conception
          comment: Typed the progress report for the presentation tomorrow to give
Dr. Page an update on our projects task and goals. Also updated the design document
to Revision I for submission tomorrow.

        - date: Feb 18, 2013
          start time: 7:23PM
          end time: 10:44PM
          phase: Coding
          comment: Worked on implementing the server email components. Took my
designs on paper from yesterday and worked them into working functions. I was able
```

```
to finish an email splitter which splits one email sent to many recipients into
multiple messages and XML output for email messages.

        -       date: February 19, 2013
                start time: 9:00am
                end time: 10:30am
                phase: Coding
                comment: Updated shell script to generate static script that will
invoke the registerUser function also defined in the register-user.lisp file.
Temporary lisp file is generated with static XML information and then redirected
into the ACL2 program in order to parse the information.  File input is delegated to
the shell, file output is delegated to ACL2.

        - date: Feb 19, 2013
          start time: 10:30AM
          end time: 11:45AM
          phase: Conception
          comment: Team Dijkstra's regular meeting time. We delivered a progress
report to Dr. Page on our task so far. Also we updated our SVN repository to the
most recent changes to the groups implementation. Isaac rejoined the group and the
remaining time was spend catching him up on the groups progress.

        -       date: February 19, 2013
                start time: 3:00pm
                end time: 4:00pm
                phase: Coding
                comment: Corrected references the xml-scanner to ignore whitespace
since some of the whitespace tokens that were being parsed were causing issues with
the interpretation of the xml tokens.  user/registration module deemed complete at
current point in time.

        - date: Feb 22, 2013
          start time: 4:32PM
          end time: 7:11PM
          phase: Coding
          comment: I worked on the ACL2 implementation of the server-email functions
and finished the implementation of the components. The functions did not admit
correctly and the details are described in the defect log. The issues were traced to
file IO so we are still able to run and test the computational functions.

        - date: Feb 22, 2013
          start time: 2:59PM
          end time: 4:16PM
          phase: Coding
          comment: I re-worked the file IO functions to accommodate the requirements
of the IO utilities functions. This fixed the majority of the admission issues for
these functions. Still the getEmail and runEmail functions are the global execution
functions still have work needed to get them to admit and run with ACL2.

        -       date: February 25, 2013
                start time: 8:00am
                end time: 9:30am
                phase: Conception
                comment: Starting in the development of the network connectivity
subsystem that will allow for the transmission of information across a network via
IP address.  Determined previously that the "nc" command in UNIX was sufficient in
order to accomplish this task  Proceeded to decompose the component and derived the
need for a separate "method invocation" language.

        -       date: February 26, 2013
                start time: 8:00am
                end time: 9:15am
                phase: Conception
                comment: Proceeded to develop a rough draft for the diagram that
would be formalized into the design document for network connectivity.  Upon further
investigation, I realized that I need to find a way in order to send a request to
the server to open a sending port after the client has acknowledged opening a
recieving port for data transmission.  This appears to be the most difficult
scenario for the server to overcome.

        - date: Feb 26, 2013
         start time: 10:30AM
         end time: 11:45AM
         phase: Coding
         comment: Team Dijkstra's regular meeting time. We brainstormed bug issues
and found a solution to the Server-Email IO problem. We will resolve multiple email
messages from the command shell and call ACL2 for each individual message.

        - date: Feb 27, 2013
          start time: 12:01PM
          end time: 12:53PM
          phase: Coding
          comment: Worked on the IO contents and designing the Test and Theorems for
the server email module. Had an idea to combine the IO into one function to see if
it fixes the IO issue I'm having with the server-email. Started implementation and
will finish tonight.

        - date: Feb 27, 2013
          start time: 9:54PM
          end time: 10:32PM
          phase: Coding
          comment: I finally have the IO fixed. With one call I can have and input
file in XML format parse and get passed to an output file. Now its time to handled
multiple XML messages in one file.

        - date: Feb 28, 2013
          start time: 10:30AM
          end time: 11:45AM
          phase: Coding
          comment: Team Dijkstra's regular meeting time. We worked on server
implementation and fixed issues with the server-email file. We split it into two
files where one contained the IO functions and the other contained the logic
functions. This was done in order to make proving theorems in ACL2 easier.

        - date: Feb 28, 2013
          start time: 4:02PM
          end time: 5:12PM
          phase: Testing
          comment: I wrote the theorem suite for the server-email functions. These
tested each function in the file at least once. This guarantees that the written
code does what it is needed to do.

        -       date: March 1, 2013
                start time: 12:35am
                end time: 1:45am
                phase: Conception
                comment: Started to develop the basis for the "Server Messaging
Language" that would overcome the need for remote method invocation.  Syntax of
which would be encapsulated in curly brackets and separated by semicolins.  Started
prototyping functions to scan for this language.  Header information would be passed
via first line in the XML transmission over the network and would be saved in a
timestamp relative to UNIX conception date (date +%s.txt).

        - date: March 2, 2013
          start time: 4:53PM
          end time: 5:34PM
          phase: Coding
          comment: I wrote a shell script that dynamically writes a lisp file with
the command to call the rwEmail function and output the files to a directory with
the clients name in the servers store folder. Each file is names msg_timestamp where
the timestamp is the actual timestamp of the message.

        -       date: March 4, 2013
                start time: 8:00am
                end time: 8:45am
                phase: Conception
                comment: Started to rough draft a client side development plan.  Wes
and I are considered the "server side" team, while we had hopes of Adam and Isaac
being the "client side" team.  Being that both Adam and Isaac were gone, Wes and I
have had to develop a plan of action toward pushing for project completion.  RMI has
been temporarily placed on hold until we can solve these issues.  Perhaps and
explanation of the RAD process and a bit of motivation would be the route to go.
Designed a flow chart for an overview of the basic functionality and what components
were complete.  Decided that I would give the presentation on the update of the
status of the project.

        - date: March 4, 2013
          start time: 11:49AM
          end time: 1:04PM
          phase: Coding
          comment: Worked on updating rwEmail. I changed the output file from being
solely passed in by the parameters to where only a timestamp is required. The output
directory is now dynamically generated to pull the <to> tag from the XML and outputs
to output each email message into a directory based on the contents of the tag and
the naming convention that was implemented earlier for each individual file.

        -       date: March 5, 2013
                start time: 7:15pm
                end time: 8:30pm
                phase: Coding
                comment:        Started development on the RMI language.  Instead of
developing this as a module, I have decided to place this in the "includes" folder
since it would more than likely be used by other modules.  Point of entry is
getActions function which would return a data structure of actions that will be
updated in the next design document.  On a side note, we can use this for user
authentication where required instead of a separate request for authentication.

        -       date: March 5, 2013
                start time: 2:30pm
                end time: 2:45pm
                phase: Testing
                comment: Tested out some shell solutions to multiple file traversal,
since Wes was stating he needed to find a resolution for this issue, as discussed in
our meeting earlier today.  Posted a short "how-to" on the for loop and file
acquisition.  Wes responded shortly after implementing the code to verify its
working condition.  It appears to be the solution we were looking for.  Considering
implemenation in other modules as well as the register-user module.

        - date: March 5, 2013
          start time: 7:24PM
          end time: 7:48PM
          phase: Coding
          comment: Worked on updating the shell script to include a for loop to
process each file that exist in the incoming directory. Also edited, the route-email
ACL2 file to change the directory structure of the email output.

        -       date: March 7, 2013
                start time: 12:00am
                end time: 2:30am
                phase: Coding
                comment: Continued development on the RMI language interpreter.
Developed last few functions getActionString, getAction.  Brackets encapsulate the
actions, and semicolins separate the actions.  SERVERACTION denotes a server action
to take place.  CLIENTACTION denotes that a client action should take place.  At the
time of this file conception, MOVEFILE and COPYFILE actions are only considered.
The portion containing the string between the brackets is the module that will be
considered.  This will acquire the "monitor" value from the module registration on
the server side.  In otherwords, the file that contains the "header" information
will be MOVED/COPIED (depending on action) to the monitor for the module that is in
the brackets, thus invoking the monitor process and kick starting the module itself.
 Files that contain monitor information will end with a file extension of
$unix_timestamp.rmi.  When this file is copied, the header information is removed
and the output is the following XML that is contained in the file
($unix_timestamp.xml).


       - date: March 7, 2013
         start time: 7:30PM
         end time: 8:30PM
         phase: Conceptual
         comment:  Creation of the create-user-request.lisp file. Planning how to
generate XML for user access requests

        - date: March 9, 2013
          start time: 7:59PM
          end time: 9:25PM
          phase: Coding
          comment: Worked on the client code for the email module. I wrote the
functions to parse strings into XML files for outgoing messages. I also handled the
need for the client to sent an email to multiple recipients to where a separate XML
file will need to be written for each recipient indicated in the to field of the
email xml.

        - date: March 10, 2013
         start time: 5:30PM
         end time: 7:30PM
         phase: Coding
         comment:  Completion of createRequests function in
create-user-request.lisp. Creation of the create-user-request.sh script. This script
invokes the create-user-request.lisp file

        - date: March 11, 2013
          start time: 12:04PM
          end time: 12:51PM
          phase: Coding
          comment: I worked on writing IO code for the client side email module.
This included writing code for both incoming and outgoing messages. Since these
needed to be handled differently, there are multiple functions for each of these
requirements.

        - date: March 11, 2013
          start time: 8:13PM
          end time: 9:18PM
          phase: Coding
          comment: I finished the IO code for the client email. It now handles
multiple recipients and will output a single XML file that will need to be processed
to send the multiple email messages. A shell script was written to handled ALL
incoming messages and process them to HTML files for easy reading.

        - date: March 12, 2013
          start time: 11:46AM
          end time: 12:49PM
          phase: Coding
          comment: The client email code has been finished. Worked on tweaks to get
the output correct and in the correct XML format. Worked on naming of files to
ensure unique naming for each generated file. Finished a shell script to split the
email output that contains multiple recipients.

        -       date: March 13, 2013
                start time: 9:00am
                end time: 10:30am
                phase: Coding
                comment: Began adding shell commands to extract the domain and name
from the registration files to be able to create the directories for the email
storage on the server.

        - date: March 13, 2013
          start time: 5:13PM
          end time: 7:14PM
          phase: Testing
          comment: Tested the client code. Worked on verification of the connection
between the logic module and the IO module. Ensured that the correct functions were
called, and returned the correct structures. This is set up for writing the theorems
for this module. Also added the getHTMLtext to the client logic to allow an HTML
file to formed and written instead of regular plain text for a nicer looking output.

        - date: March 14, 2013
          start time: 6:43PM
          end time: 7:34PM
          phase: Coding
          comment: Wrote the shell scripts to automate the client email processing.
One shell script was made to handle incoming email messages and directs incoming
messages to the inbox folder. The other shell script handled outgoing messages and
places them in the outbox folder.

        - date: March 15, 2013
          start time: 9:46PM
          end time: 10:52PM
          phase: Coding
          comment: Added to the outgoing shell script the capability to parse
multiple emails and create an XML file that contains a single email message with a
unique file name. This shell script will then send the single email script to the
server using the nc command.

        -       date: March 17, 2013
                start time: 12:15am
                end time: 12:48am
                phase: Coding
                comment: Finished adding the directory creation mechanisms to the
shell scripts and tested the user registration process.  It seems to be working
correctly now.

        -       date: March 17, 2013
                start time: 12:50am
                end time: 12:55am
                phase: Testing
                comment: Continued to test the functionality of the user
registration process on the server side.  One thing to note, we cannot have spaces
in our name and domain.  Whitespace has been trimmed.

       - date: March 17, 2013
         start time: 9:30PM
         end time: 10:00PM
         phase: Coding
         comment:  Changed parameters of createRequests function to accept an
```

```
argument for the time stamp of creation used by the create-user-request.sh script

        -       date: March 17, 2013
                start time: 1:00am
                end time: 1:15am
                phase: Coding
                comment: Added and modified lines in the register-user and
address-book lisp files to allow for a user to register with a password.  This will
be used to verify the user can perform actions on the server side.

        -       date: March 17, 2013
                start time: 1:15am
                end time: 3:00am
                phase: Coding
                comment: Created the verify action on the user module.  Defined
utility functions to acquire domain, name, password and the location of the client.
The location will need to be extracted from the original XML from the shell script
since it does not conform the the verification user information.

        -       date: March 17, 2013
                start time: 3:00am
                end time: 6:17am
                phase: Coding
                comment: Created all the functions that will perform the actions on
the mailing list.  subscribe, unsubscribe, and all predicates to allow for these
actions to complete.

        -       date: March 17, 2013
                start time: 6:37am
                end time: 6:45am
                phase: Testing
                comment: After looking at some of the properties from the
address-book_tests file, I realize that I forgot to log these functions into the PSP
logs from last cycle (probably because I did them right before class and lost track
of time).  These have been added to this log.

        -       date: March 23, 2013
                start time: 2:00am
                end time: 2:45am
                phase: Coding
                comment: Added verify-user.sh file to invoke the actions required to
start the user verification process on the server side.  Having a few issues getting
remote-actions.lisp to be accepted into logic (guard checking issues).

        -       date: March 23, 2013
                start time: 12:00am
                end time: 2:50am
                phase: Coding
                comment: Finished the verify-user.sh action file for establishing a
connection between the client and server for mail reception.

        - date: March 24, 2013
          start time: 4:58PM
          end time: 6:18PM
          phase: Testing
          comment: Worked on the theorem suite for the client email module. The
theorems are used to test the data integrity of the logic functions in the client.
We were unable to test the IO functions with theorems since they rely on variant
data. However, the logic could be tested using theorems and we were able to prove
that the client email logic module returns the correctly formatted data based on
correct input.

        - date: March 25, 2013
```

```
          start time: 6:12PM
          end time: 6:49PM
          phase: Coding
          comment: Tweaked the shell scripts for the clients to ensure that the
server has ample time to process a single email message and that the client will not
overload the server by adding too many emails to the queue.

       - date: March 25, 2013
         start time: 7:30PM
         end time: 8:30PM
         phase: Conceptual
         comment:  Creation of the create-block-request.lisp file.  Planning how to
generate XML for requests to block users

       - date: March 25, 2013
         start time: 9:30PM
         end time: 11:00PM
         phase: Coding
         comment:  Completion of the function responsible for generating xml for
creating requests to block users as well as I/O functions

       - date: March 25, 2013
         start time: 11:30PM
         end time: 11:59PM
         phase: Coding
         comment:  Creation of create-block-request.sh script.  This script invokes
the create-block-request.lisp file

        - date: March 27, 2013
         start time: 9:30PM
         end time: 10:30PM
         phase: Coding
         comment:  Creation of create-mailing-list.lisp file. Creation of the XML
Generating functions addressesXML and ownerXML

        - date: March 28, 2013
         start time: 7:00AM
         end time: 8:00AM
         phase: Coding
         comment:  Creation of file output method

        - date: April 1, 2013
          start time: 5:55PM
          end time: 6:47PM
          phase: Coding
          comment: Worked on reworking the Java Gui client to work directly with
ACL2 instead of relying on shell scripts. This allows faster and more secure
connections between remote host for our networking portions.

        - date: April 1, 2013
          start time: 8:59PM
          end time: 11:10PM
          phase: Coding
          comment: Continued to work on the Java integration with ALC2. I was able
to complete most of the Send email functions for the client. All that remains on
this front is sending the file's contents over the network.

        - date: April 2, 2013
          start time: 6:32PM
          end time: 9:48PM
          phase: Coding
          comment: I reworked the client actions into the required Java programs.
They now work with the server and can send and receive information. The scripts that
```

```
handled the ACL2 function calls were also integrated into the GUI interface and are
easily accessible to the user if they are running the interface.

        - date: April 2, 2013
          start time: 9:49PM
          end time: 10:04PM
          phase: Testing
          comment: I found a bug with the server code that was already implemented.
When looping though all files in a directory, it was picking up extra hidden files
as well, This caused the server to crash. I added code to fix this issue.

        - date: April 3, 2013
          start time: 5:47PM
          end time: 8:11PM
          phase: Coding
          comment: I finished coding the Java integration for the current ACL2
implementation. The client side of the program can now send and receive emails and
User registration. This also includes user verification in order to get email
messages.

        - date: April 4, 2013
          start time: 4:31PM
          end time: 5:34PM
          phase: Coding
          comment: I added the delete function to the client GUI interface. I also
worked on fixing a bug on the transmission of messages. The current issue is that
the verification module does not allow for correct exceptions to be processed from
the server. If a transmission fails, it does not handle the output correctly and
sends an incorrect XML file back to the client.


defect log:

        -       date: January 18, 2013
                type: Networking
                fix time: 60
                comment: IP resolution cannot occur in ACL2 unless a network drive
is mapped, after which you can call it by its network path.  Networking drives may
be our resolution to this issue.

        - date: Jan 22, 2013
          type: Design
          fix time: 60
          comment: Found out that networking is not feasible from within ACL2. To
make it natively supported, writing and extending several Common Lisp features would
need to be done. We cannot do this in the scope of this project. So we found that
using the Operating System's native filesystem and networking support would be much
more friendly to deal with once we get to this stage in the project.

       - date: Jan 29, 2013
         type: Design
         fix time: 147
         comment: Found out that GUI's and File IO cannot coexist in ACL2. We can
have one or the other, but not both. So we had to scrap the GUI portions of the
project and replace them with a new idea. The new idea is the current design of the
email server and client system. This project is strictly data processing and file
IO. This project will be file and text based rather than Visual and Interactive.

        -       date: January 29, 2013
                type: Conception
                fix time: 150
                comment: ACL2 cannot work with a GUI and IO at the same time.  Had
to reevaluate how to salvage what we had regarding design.  Instead of real time
```

chat, we would be sending email. Instead of ACL2 interfaces, we would program them in C or C++.

- date: January 31, 2013
  type: Application Support
  fix time: 90
  comment: Unable to get Proofpad to work correctly on any of my computers. Windows 8 and Ubuntu both show NullPointerExceptions when trying to implement ACL2 and Proofpad will not correctly identify with ACL2. Opted to use Dracula instead.

- date: February 3, 2013
  type: Conception
  fix time: 150
  comment: Determined that the size of the executables generated by ACL2 would not be a practical application for our program. Opted to use input redirection into ACL2 prompt and shell script invocation. Ubuntu would be the server platform and the two Macs would be used as clients to send and recieve information.

- date: February 5, 2013
  type: Personnel
  fix time: 15
  comment: With the loss of Isaac from the team, modules had to be prioritized for completion and new due dates had to be set. Determined that I would need to finish the XML Parser as quickly as possible to begin development in order to maintain deadlines.

- date: Feb 5, 2013
  type: Design
  fix time: 124
  comment: We discovered that generating ALC2 executables and invoking these files from an outside source is a troublesome experience and that the generated files are hundreds of megabytes in size. Since we will have several modules for this project, we saw this as a negative side effect of executable files. To solve this problem, we moved all our project to the UNIX platform. This has allowed us to use the UNIX shell environment to generate shell scripts that invoke the ACL2 environment while passing in ACL2 source code files. This reduces the size of the files to kilobytes and streamlines the execution process and eliminating the size of outside programming needed for the original idea to work. Thus we will be executing our ACL2 code through a UNIX shell script and the shell scripts will in turn be executed from the outside programming environment.

- date: Feb 12, 2013
  type: Design
  fix time: 75
  comment: When we looked at the design review. We saw several errors in the XML format that would not pass if it were to be sent though a web browser. We had to work on setting the XML to a correct format and modify the document type definitions to comply with proper XML syntax.

- date: February 15, 2013
  type: Coding
  fix time: 10
  comment: Made the decision to allow shell scripts to take care of much of the IO on the read side as possible and file operations would need to be done by the OS in order to keep correct RWE privs on the file for security purposes. CHMOD properties will need to be determined at a different date, since access to store files has not been completely determined. Best guess is that server will be the only one that needs read/write access to these files and no user group will need execution access.

- date: February 19, 2013

```
                type:   Coding
                fix time: 20
                comment: Issue arose when parsing XML tokens that the whitespace in
the document was being identified by #PCDATA token, which was incorrect.  This
created a case where the next token to be identified was not was predicted thus
returning a nil result when processing the address-book xml input.  Created a
special case that modified around 8 lines of code to check if there exists a
whitespace character outside of encapsulating brackets and if so, to ignore those
values.  Had to remove 4 lines of old predicate for prediction of the next token to
be < character.

- date: Feb 22, 2013
         type: Coding
         fix time: 12
         comment: After finishing coding the ACL2 functions, the functions would not
admit under normal ACL2 invocation. However, it did work under Dr Racket. The issue
was traced to the IO and List utilities files as they were un-certified files within
the regular ACL2 environment. Adding the suppression to the certification
requirement, the files worked as usual.

- date: Feb 23, 2013
         type: Coding
         fix time: 52
         comment: After fixing the certification issue. Certain functions were still
not admitting to ACL2. This was due to illegal arguments as the ACL2 output stated.
To fix these arguments, the state variable had to be set and implemented differently
than I had intended. I added some safe guards to the variables and added constraints
to the functions that depended on them.

        -       date: March 4, 2013
                type: Personnel
                fix time: 45
                comment: Has to reconsider a new "plan of action" with regards to
completion of the project.  Current methods seemed to archaic for the current
situation as we were unable to adapt due to the relaince on people.  Opted for a RAD
development solution where SCRUM and Extreme Programming were the foundations for
development.  Will need to explicitly sit down and speak with team to describe the
course of action.  Developed a visual aid to describe where we are and what is
completed.  There seems to be more questions on this as opposed to the completion of
the project.

-date: March 5, 2013
        type: Coding
        fix time: 21
        comment: After finishing the IO entry point function on the server email
module, We noticed that the XML files were being generated with a comma instead of
the @ symbol between names and domains in the email address. Also, the output file
for the email was a statically named file. This file needed to be dynamically named
with a timestamp.

-date: March 11, 2013
        type: Coding
        fix time: 24
        comment: Email parsing had a one off error that did not have the correct
lines returned for the XML file which rendered the outputted file useless.

        -       date: March 13, 2013
                type: Coding
                fix time: 20
                comment: Was not able to get email to write to the server.
Determine that it was an issue that the directories did not exist, thus the
information was not being written properly by the ACL2 script, which was a difficult
thing to determine since acquiring error information from the runtime environment
```

```
would only be possible if we wrote the error to an output log.  Determined that I
would need to finish the registration process by adding directory creation to the
shell script in order to make this function correctly.  Until then, we can manually
create the directories.

-date: March 14, 2013
        type: Coding
        fix time: 32
        comment: Shell script was not correctly splitting the XML files based on the
regular expression. Started using awk to parse the file. However, the file did not
have a unique file name and was getting overwritten every time the script was
executed.

        -       date: March 23, 2013
                type: Coding
                fix time: 35
                comment: Type checking for string-listp on XML conversion in the
server actions was incorrect.  Was using endp and stringp tests, when combined I
could use string-listp, which ended up being the required fix and not having to turn
off guard checking.

-date: March 24, 2013
        type: Testing
        fix time: 16
        comment: The theorem that tested the email data structure was not passing.
This was traced to an error in the proof and not in the code. The error was trying
to access an item that was not in the structure, hence the failure of the proof.

        - date: Mar 27, 2013
          type: Design
          fix time: 00
          comment: Need to restructure the create-mailing-list.lisp file to handle
multiple addresses and multiple owners.

-date: April 1, 2013
        type: Coding
        fix time: 23
        comment: Working on integrating the ALC2 with Java. Having trouble getting
Java to see the files that ACL2 has generated. Right now, the current solution is to
make the Java sleep for a couple of seconds while ACL2 finishes its processing then
resume. Then it sees the files that ALC2 generates

-date: April 2, 2013
        type: Testing
        fix time: 15
        comment: The script that Matthew had written to open all files in a
directory was not working. It was needed that the function open only the xml files,
since there are hidden files in a directory, I had to modify the function to account
for these changes.
```