$(\text{mail})^n$

Elaboration II

CS616
Fall 2004
Dr. Marchese
December 1, 2004

Elias Rosero
Jwalant Dholakia
Joseph Aulisi

Elaboration Phase II

The second part to the elaboration process concentrates on three artifacts that were also mentioned in earlier documentation. The three artifacts are *Domain Model (refined), SSDs (expanded),* and *System Contracts (expanded)*.

Phase II covers many parts of the system that were not studied in earlier stages of development. For example, in the System Contracts there are additional contracts that describe functions of (mail)^n that were not discussed before. Examples of those functionalities are the ability of an employee to create a *contacts* list, and to create *folders* (for organizing emails). In addition, some administrative functions are presented as well.

These functionalities, although important to the system, were considered as features of the system as a whole. In Phase I, we focused on the core functionalities of (mail)^n. However, as we continue with the development process, it has become necessary to study the secondary functionalities of (mail)^n.

During this iteration, the artifacts are revised and augmented. The changes are described with detail in the rest of this document. However, we will keep in mind that there is yet another phase of the elaboration process that follows this one. And as such, the material presented in this document is subject to modifications in the near future.

Refined Domain Model

Some modifications have been made to the domain model as a part of Elaboration II. These changes are as under:

1. A new conceptual class of (mail)^n Database has been recognized and its associations with the Mail Server and (mail)^n Website have been established.

2. Information regarding the Search functionality has been incorporated in the Domain Model by creating an association between User and Mail Conceptual Classes.

3. User mail has been classified into sub-categories including inbox, sent, erase and new_folder(s).

4. User Groups has been specified as a separate Conceptual Class, which can be created by users.

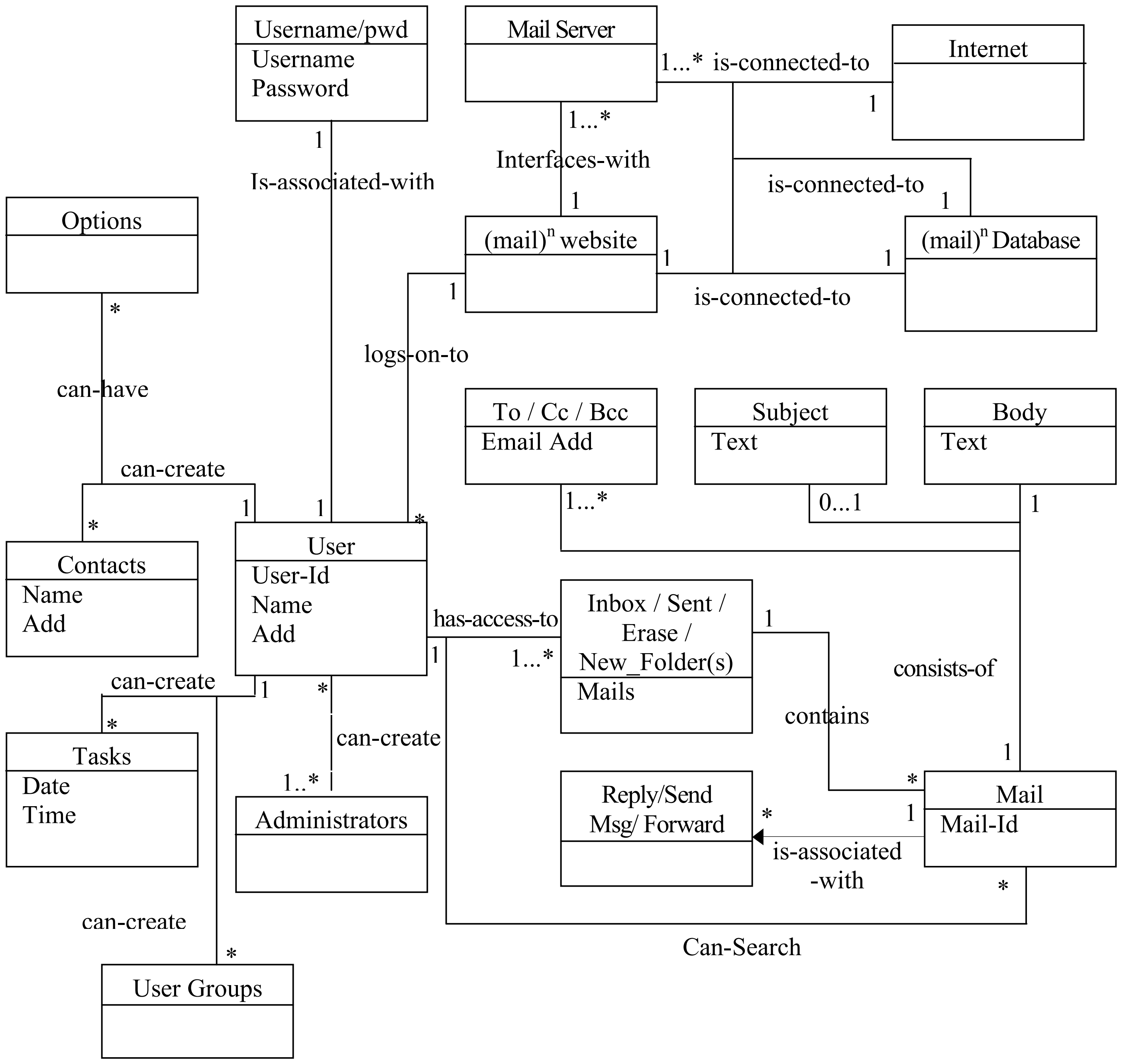

Figure 2.0 Refined Domain Model

SSDs Expanded

In the Elaboration Phase I document, we included a single SSD diagram that *translated* the use-case scenario of a happy path through the system as described in the early stages of development. The diagram was used to clarify the contents of the *Processing Message* use-case. However, the SSD presented in Phase I is incomplete. It was used solely as a supplementary source of information for the textual use-case presented.

For Phase II of the Elaboration, we will expand the SSD contents of the *Processing Message* use case. In addition, we will also present expanded versions of SSDs that describe the rest of the functionalities of (mail)[n], namely, the functionalities of employee options and administrator options.

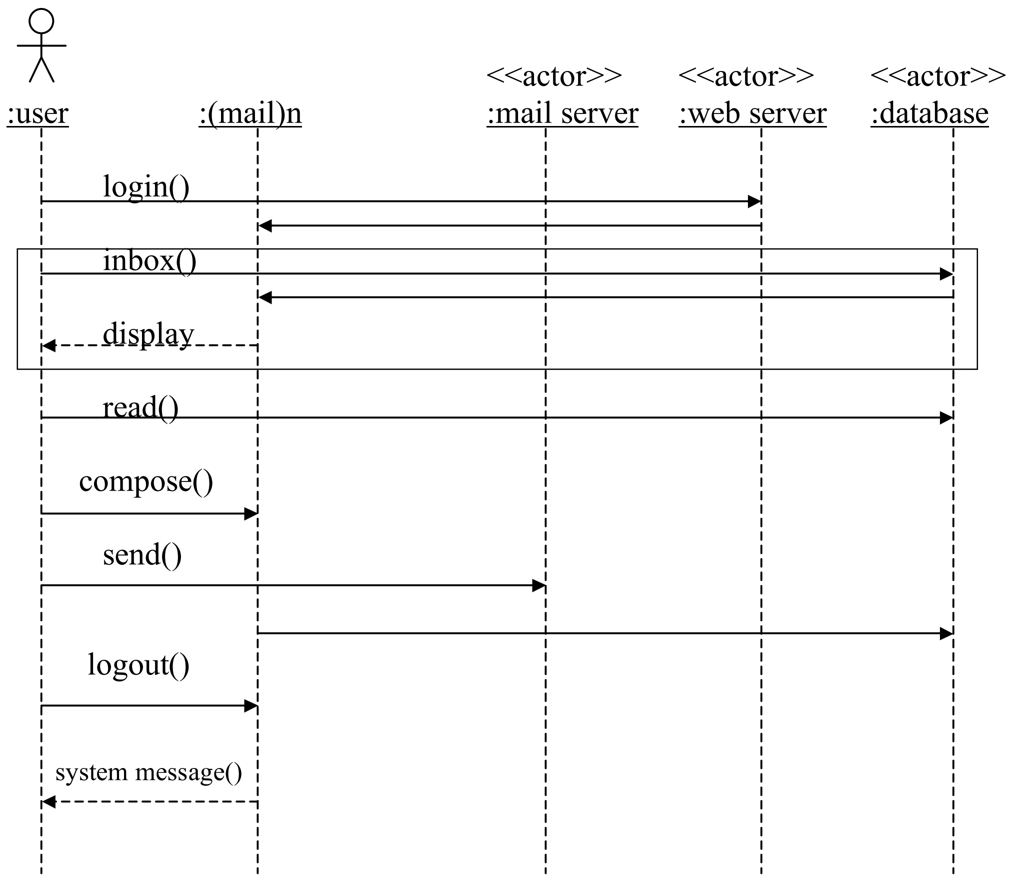

Figure 1.0 SSD of Processing Messages

In figure 1.0 we observe an SSD that graphically represents what the use-case scenario presented as text in the earlier stages of development described. The graph contains the user, the system (mail)[n], and the three actors: web server, mail server, and database. The actions that stem from the user are represented with vectors that expand to the appropriate 'instance'. For example, when the user wants to send an email message, upon a click of the mouse or a keyboard entry, first the action is processed through the web server, and then the system saves a copy of the message in the database. The first action is shown with a vector from the user to the web server, illustrating that the user has requested to send a message. Directly below, there is another vector originating from the system (mail)^n pointing to the database to illustrate that the system saves a copy.

It is necessary to recall that SSDs are drawn with a chronological order downward. Which means that the actions are sequential. It is also important to clarify that the actions that are contained within the box are those actions that are iterative.

The second SSD is a graphical representation of the use-case "Employee Options". Although the SSDs follow a chronological order, in this case the actions taken by the user/employee are specified in a random order. The options listed are accessible to the user at any point within the system. The actions are therefore listed without an order.

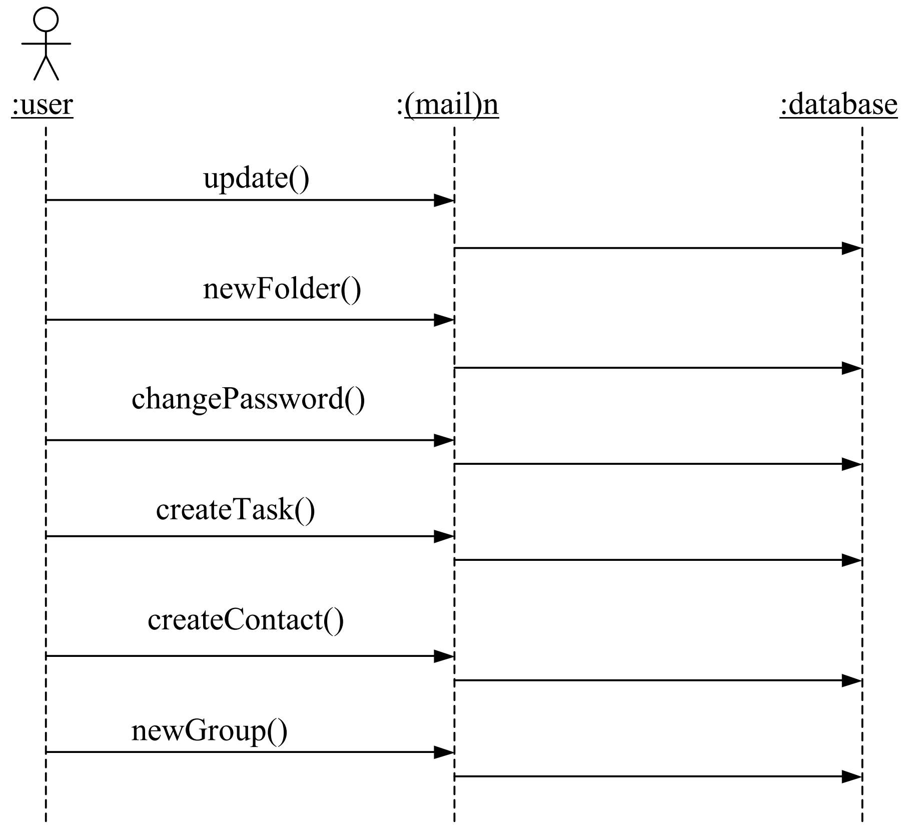

Figure 1.1 SSD of Employee Options

In figure 1.1 we have and SSD that shows the various options that an employee has within the system. There are no responses made to the user that are necessary to point out, because the updates are made to the database. For example, if a user wants to create a new folder to organize his contacts into groups, then the user clicks on the link for creating new folders and the system automatically prompts for the name of the new folder. Once the user clicks on OK, the request is performed in the database.

The following SSD represents the use-case designed for the Administrator's Options. As stated in early documentation, the administrator has privileges that a normal does not have. For example, an administrator has the power to create, deleted or modify accounts. The initial screen is different for the administrators.

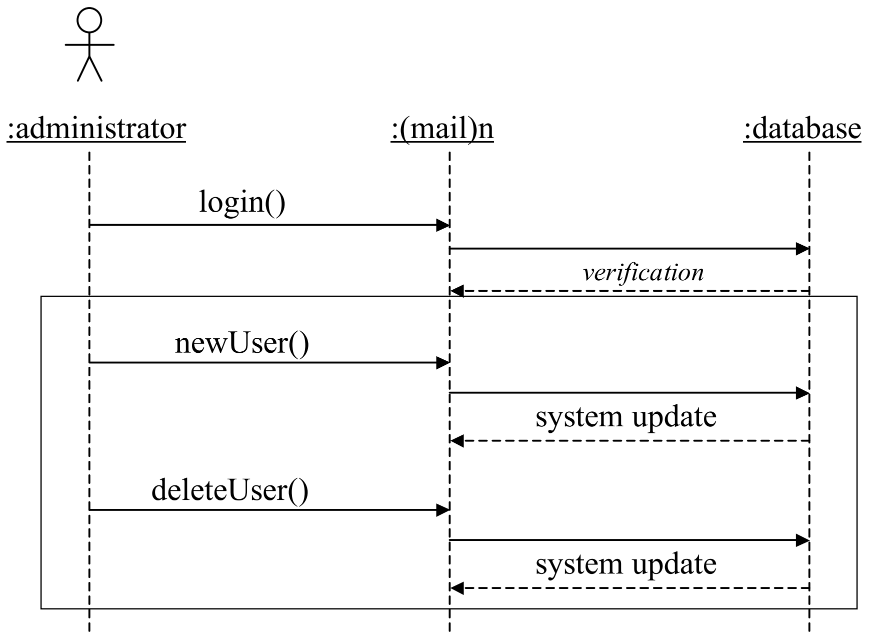

Figure 1.2 SSD of Administrator Options

Upon verifying that an administrator has logged on to the system, the administrator may add or delete users. In figure 1.2 the administrator's options are represented in the diagram. An administrator, when adding a user to the system, is basically updating the *employee table* as well as the *login table.*

As per any of the SSDs in this section, they may be further clarified by reading the *Use case system contracts*. Within the contracts, the tables that are updated upon an action are explicitly mentioned.

Use Case System Operations Contracts

Contract C1: send

Operation:	send(message-id, e-id, recipients)
Cross References:	Use Case UC1: Processing Messages
Pre-Conditions:	At least the "To" field is filled with a valid email address.
Post-Conditions:	- a unique message id was created - the message id was associated with the current e-id - a connection with the mail transfer agent was made - the message was saved to the mail database, and an entry was made in the sent table

Contract C2: fetch

Operation:	fetch(message-id, e-id, recipients, authenticator)
Cross References:	Use Case UC1: Processing Messages
Pre-Conditions:	Messages exist on the mail server.
Post-Conditions:	- an authentication was made associated with creating a session - an connection with IMAP was made - the message was stored in the mail table - an entry was made in the inbox table - an associated entry was made in the flag table - the read flag in the flag table was marked as false - the session was closed

Contract C3: read

Operation:	read()
Cross References:	Use Case UC1: Processing Messages
Pre-Conditions:	There is a message to read.
Post-Conditions:	- the read field of the flag table was marked as true

Contract C4: reply

Operation:	reply(message-id, e-id, recipients)
Cross References:	Use Case UC1: Processing Messages
Pre-Conditions:	At least the “To” field is filled with a valid email address. There is a message that has been received.
Post-Conditions:	- a unique message id was created - the message id was associated with the current e-id - a connection with the mail transfer agent was made - the message was saved to the mail database, and an entry was made in the sent table - the associated entry in the flag table for the original message was updated with the reply attribute set to true

Contract C5: forward

Operation:	forward(message-id, e-id, recipients)
Cross References:	Use Case UC1: Processing Messages
Pre-Conditions:	At least the “To” field is filled with a valid email address. There is a message that has been received.
Post-Conditions:	- a unique message id was created - the message id was associated with the current e-id - a connection with the mail transfer agent was made - the message was saved to the mail database, and an entry was made in the sent table - the associated entry in the flag table for the original message was updated with the forward attribute set to true

Contract C6: delete

Operation:	delete(message-id, e-id)
Cross References:	Use Case UC1: Processing Messages
Pre-Conditions:	There is a message to be deleted.
Post-Conditions:	- the message-id and e-id were stored in the erase table - the associated entry was removed from the inbox table

Contract C7: empty

Operation:	empty(message-id, e-id)
Cross References:	Use Case UC1: Processing Messages
Pre-Conditions:	There is a message to be deleted.
Post-Conditions:	- the message-id and e-id were first removed from the erase table due to table constraints - the associated entry was removed from the flag table, due to table constraints - the entry was then removed from the mail table

Contract C8: move

Operation:	move(message-id, e-id)
Cross References:	Use Case UC1: Processing Messages
Pre-Conditions:	There is a message to be moved.
Post-Conditions:	- the message-id and e-id were stored in the new_folder table - the associated entry was removed from the inbox table

Contract C9: update

Operation:	update(form_data)
Cross References:	Use Case UC2: Employee Options
Pre-Conditions:	An individual has a valid account.
Post-Conditions:	- the employee table was updated with the correct data - the updated information was associated with the current e-id

Contract C10: newFolder

Operation:	newFolder(folder_name)
Cross References:	Use Case UC2: Employee Options
Pre-Conditions:	An individual has a valid account.
Post-Conditions:	- the employee table was updated with the correct data - the updated information was associated with the current e-id

Contract C11: changePassword

Operation:	changePassword(password)
Cross References:	Use Case UC2: Employee Options
Pre-Conditions:	An individual has a valid account.
Post-Conditions:	- the login table was updated with the correct data - the updated information was associated with the current e-id

Contract C12: createTask

Operation:	createTask(date, time)
Cross References:	Use Case UC2: Employee Options
Pre-Conditions:	An individual has a valid account.
Post-Conditions:	- an entry was made in the tasks table with the date and time parameters - the entered information was associated with the current e-id

Contract C13: createContact

Operation:	createContact(form_data)
Cross References:	Use Case UC2: Employee Options
Pre-Conditions:	An individual has a valid account.
Post-Conditions:	- a new entry was made in the contacts table - the entered information was associated with the current e-id - a field was set denoting the entered contact as public or private

Contract C14: newGroup

Operation:	newGroup(group_name, group_members)
Cross References:	Use Case UC2: Employee Options
Pre-Conditions:	An individual has a valid account.
Post-Conditions:	- an entry was made in the groups table - the entry was associated with the e-id of the person creating the group

Contract C15: newUser

Operation:	newUser(e-id, password)
Cross References:	Use Case UC3: Admin Options
Pre-Conditions:	An individual has a valid administrative account.
Post-Conditions:	- a new entry was made in the login table - this entry was associated with a new entry in the employee table, which was also entered at this time

Contract C16: deleteUser

Operation:	deleteUser(e-id)
Cross References:	Use Case UC3: Admin Options
Pre-Conditions:	An individual has a valid administrative account.
Post-Conditions:	- the entry was removed from the login table - the associated record was removed from the employee table

Contract C17: staffMeeting

Operation:	staffMeeting(date, time)
Cross References:	Use Case UC3: Admin Options
Pre-Conditions:	An individual has a valid administrative account.
Post-Conditions:	- an entry was made in the tasks table for each employee

Contract C18: publicGroup

Operation:	publicGroup(group_name, group_members)
Cross References:	Use Case UC3: Admin Options
Pre-Conditions:	An individual has a valid administrative account.
Post-Conditions:	- an entry was made in the groups table - the entry was associated with all employees