Leopold–Franzens–Universität
Innsbruck

Institut für Informatik
Forschungsgruppe DPS
(Distributed and Parallel Systems)

Benutzerunterstützte Emailreduktion

Bachelorarbeit

Betreuer: Dr. Michael Welzl

Thomas Nolf (0315328)

Thomas.Nolf@student.uibk.ac.at

Innsbruck
19. Mai 2008

Zusammenfassung

Das Kommunikationsmedium Email genießt heute eine weite Verbreitung und vielseitige Anwendung. Dabei kommt es im alltäglichen Umgang mit diesem Medium leider immer wieder vor, dass Emails versendet werden, die man nachträglich lieber nicht abgesendet hätte. Sei es, dass der Inhalt unvollständig ist, Anhänge fehlen oder sich herausstellt, dass der Empfänger längere Zeit abwesend ist und das Email dadurch nicht mehr relevant ist.
In dieser Arbeit wurde eine Lösung entwickelt, die ein Zurückziehen von bereits gesendeten Nachrichten erlaubt und dadurch zur Reduktion von Emails beitragen soll.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Email (Electronic Mail) ist ein schnelles, asynchrones Medium um zu kommunizieren. Die Verwendung dieses Mediums ist weit verbreitet und die Anzahl der Benutzer steigt stetig an. Dabei kommt es im alltäglichen Umgang mit diesem Medium leider immer wieder vor, dass Emails versendet werden, die sich nachträglich als unnötig herausstellen. Sei es, dass der Empfänger einer Nachricht auf Urlaub ist und der Inhalt nach seiner Rückkehr nicht mehr aktuell bzw. nicht mehr von Bedeutung ist, oder dass Emails verfrüht abgesendet werden und dadurch Inhalt wie Text oder Anhänge fehlen. Der Absender muss meist ein zweites Email nachsenden um das Erste zu korrigieren bzw. zu vervollständigen. In der Zwischenzeit kann es auch vorkommen, dass sich der Empfänger über die fehlenden Informationen erkundigt, was zusätzlichen Emailverkehr erzeugt.

Ziel dieser Arbeit ist es, Benutzern die Möglichkeit zu geben Nachrichten nachträglich zurückziehen zu können um damit zur Reduktion von Emails beizutragen. Dies soll durch zwei verschiedene Ansätze ermöglicht werden: zum Einen soll bei Abwesenheit des Empfängers eine Urlaubsnachricht an den Absender versendet werden welche auf die Abwesenheit hinweist. Durch Antworten auf diese Abwesenheitsnachricht soll der Absender die Möglichkeit haben das ursprüngliche Email wieder zu löschen. Zum Anderen soll eine Möglichkeit geschaffen werden jedes beliebige Email zurückziehen zu können, welches vom Empfänger noch nicht vom Postfach geladen wurde.

In Kapitel 2 werden bestehende Ansätze analysiert und bewertet. Es folgt ein Kapitel über die Anforderungen an die Lösung und über deren Umsetzung. Es werden die verwendeten Technologien genannt und die Struktur der Programme sowie deren Funktionsweise erläutert. In Kapitel 4 wird dargestellt, wie die einzelnen Lösungen getestet wurden. Abschliessend gibt es in Kapitel 5 eine Zusammenfassung der verwendeten Software und einen Rückblick. Darüberhinaus gibt es im Anhang ein Readme, welches die Installation und Verwendung der in dieser Arbeit erstellten Programme erklärt.

2 Analyse bestehender Ansätze

Auf der Suche nach Möglichkeiten zum Zurückziehen von bereits gesendeten Emails wurden mehrere Lösungen gefunden, die man grob in zwei Kategorieren einteilen kann:

- Lösungen als Bestandteil von Emailclients
- Lösungen als unabhängige Online-Dienste

Die erste Art ist nur beim Emailclient Microsoft Outlook bekannt, der ein solches Feature seit Microsoft Office Outlook 2000 enthält. In der zweiten Kategorie gibt es verschiedene Online-Dienste, die Emails auf ihren Servern zwischenspeichern und durch verschiedene Techniken ein Zurückziehen der Emails erlauben.

2.1 Recall durch Microsoft Outlook

Recall bezeichnet jenes Feature von Microsoft Outlook [1], welches das Zurückziehen von Emails über eine Funktion des Emailclients erlaubt. Dieses Feature kann nur benutzt werden, wenn sowohl der Sender als auch der Empfänger des Emails diesen Emailclient und einen Microsoft Exchange Server verwenden.

Um ein bereits gesendetes Email wieder zurückzuziehen, muss dieses geöffnet werden. Anschliessend kann man im Menü "Aktionen" den Punkt "Diese Nachricht zurückrufen..." aufrufen. In einer Dialogbox (siehe Abbildung 2.1) kann man auswählen, ob die gesendete Nachricht beim Empfänger gelöscht oder durch eine neue Nachricht ersetzt werden soll. Weiters kann man sich über den Erfolg oder Misserfolg des Nachrichtenrückrufes informieren lassen. Nach Bestätigung wird ein Recall-Email mit einer Datei winmail.dat versendet, die dann beim Empfänger den Rückzug auslöst.

Wurde das Email vom Empfänger zum Zeitpunkt des Rückzuges bereits gelesen, so bekommt er die Nachricht, dass der Absender das Email löschen wollte. Das Email bleibt jedoch in seiner Mailbox. Wurde das Email noch nicht gelesen, so wird es gelöscht und der Empfänger darauf hingewiesen, dass der Absender sein Email wieder zurückgezogen hat.

Ist die Einstellung "Automatische Bearbeitung beim Eintreffen" beim Emailclient des Empfängers nicht aktiviert, so kommt das Recall-Email im Postfach wie jedes andere Email an und das Verhalten hängt davon ab, ob zuerst das Recall-Email oder das ursprüngliche Email geöffnet wird.

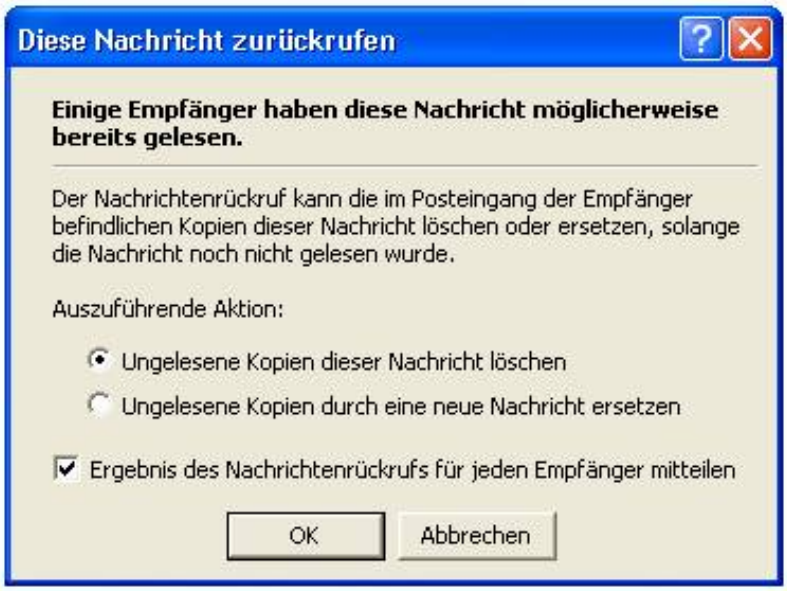

Abbildung 2.1: Nachricht zurückrufen bei Microsoft Outlook

Wird zuerst das Recall-Email vom Empfänger geöffnet, so wird das ursprüngliche Email gelöscht und es folgt der Hinweis, dass das Email vom Absender gelöscht wurde. Im zweiten Fall schlägt das Rückziehen fehl und beide Emails bleiben im Postfach des Empfängers.

2.2 Recall durch Online-Dienste

In diesem Abschnitt werden die zwei Online-Dienste www.bigstring.com und www.yankback.com untersucht, die das Rückziehen von Emails durch zwei verschiedene Ansätze ermöglichen.

2.2.1 Recall durch www.bigstring.com

www.bigstring.com [2] ist ein in New York ansässiges Unternehmen, dass seinen Kunden revolutionäre Email-Dienste verspricht. Es bietet einfache Möglichkeiten zum Senden, Zurückziehen, Löschen, Selbstzerstören und Abändern von Emails die bereits gesendet und eventuell auch schon gelesen wurden. Der Benutzer kann alle diese Dienste kostenlos über ein Web-Interface nutzen. Gegen Bezahlung von Gebühren wird einem auch ein Pop3-Zugang gewährt und es können Emailadressen der eigenen Domain verwendet werden.

www.bigstring.com verfolgt einen einfachen Lösungsansatz um Emails wieder zurücknehmen zu können. Dabei wird aus jeglichem Inhalt eines Emails ein Bild generiert und in das Email so eingebunden, dass es der Mailclient des Empfängers von einem Webserver laden muss um es anzeigen zu können.

Will nun der Absender sein Email zurückziehen, so wird das Bild einfach vom Server gelöscht und der Empfänger sieht nur mehr ein leeres Email ohne Inhalt.

2.2.2 Recall durch www.yankback.com

www.yankback.com [3] ist ein weiterer, kommerzieller Anbieter eines Dienstes zum Zurücknehmen von Emails. Das vom Unternehmen bezeichnete "Email Recovery System" verfolgt einen anderen Ansatz und hält das abgesendete Email eine bestimmte Zeit lang auf dem Mailserver zurück, bevor es endgültig an den Empfänger versendet wird. In diesem Zeitraum ist es dem Benutzer erlaubt sein Email über ein Web-Interface zurückzuziehen.

2.3 Zusammenfassung bestehender Ansätze

Zusammenfassend kann man sagen, dass keiner der untersuchten Ansätze eine zufriedenstellende Lösung darstellt. Die proprietäre Lösung von Microsoft ist nur für den hauseigenen Emailclient Outlook in Verbindung mit einem Microsoft Exchange Server angedacht. Ausserdem wird der Empfänger über jede versuchte Nachrichtenrücknahme informiert, was wohl die Aufmerksamkeit unnötig auf das zurückgezogene Email lenkt.

Die Lösung von Bigstring kann zwar unabhängig vom Emailclient benutzt werden, jedoch sind auch hier einige Nachteile zu nennen: so fallen für die sinnvolle Nutzung von Bigstring durch einen Emailclient Gebühren an. Ausserdem erschwert die Repräsentation des Emails durch Bilder den Umgang mit dem Inhalt. So kann der Empfänger keinen Textinhalt aus dem Email kopieren oder seine Bemerkungen bzw. Antworten zwischen die Zeilen ("inline") schreiben. Weiters ist nach der Zurücknahme noch das leere Email mit dem Betreff im Postfach des Empfängers und sorgt vermutlich für Verwunderung oder führt sogar zu Verwirrung und Nachfragen.

Auch die dritte Lösung von Yankback ist kostenpflichtig. Hier wird zwar das Email unverändert gesendet, dafür ist aber eine Zurücknahme nur innerhalb einer bestimmten Zeit möglich.

2.4 Ausblick auf die neue Lösung

Die in dieser Arbeit vorgestellte Lösung überzeugt in ihrer einfachen Handhabung und erreicht das Ziel, unabhängig vom verwendeten Emailclient eine Zurücknahme von Emails zu ermöglichen, wobei sich daraus keinerlei Vorgaben für den Aufbau der rückzugsfähigen Emails ergeben. Ein weiteres Merkmal ist, dass Recall-Emails unabhängig vom Erfolg oder Misserfolg immer gelöscht werden und dadurch keine Unsicherheit, Verwunderung oder ein erhöhtes Emailaufkommen entsteht.

3 Beschreibung der Lösung

In diesem Kapitel wird die umgesetzte Lösung von den Anforderungen bis hin zur Implementierung beschrieben.

3.1 Anforderungen

Wie in Kapitel 1 beschrieben soll das System auf zwei unterschiedliche Arten das Zurückziehen von Emails erlauben. Einerseits soll dies durch das Antworten auf eine Abwesenheitsemail geschehen und andererseits soll jedes Email durch erneutes Weiterleiten des gesendeten Emails zurückgezogen werden können. Aus diesem Grund gibt es für die beiden Anforderungen zwei unabhängige Lösungen. Das Programm für die erste Art (Antwort auf Abwesenheitsnachricht) heißt *Vacation* und jenes für die zweite (Zurückziehen jeglicher Mails) wird als *SpontaneousRecall* bezeichnet.

3.1.1 Anforderungen an Vacation

Die Lösung *Vacation* soll direkt am Mailserver betrieben werden und ähnlich dem Programm vacation.pl eine Abwesenheitsnachricht verschicken können. Dabei soll das Abwesenheitsemail dieser Variante aber einen eindeutigen Schlüssel enthalten aufgrund dessen ein sicheres Löschen des ursprünglichen Emails ermöglicht wird. Weiters soll sich der Absender durch eine Bestätigung über den Erfolg oder Misserfolg der Zurücknahme informieren lassen können.

3.1.2 Anforderung an SpontaneousRecall

SpontaneousRecall soll sowohl auf einem Mailserver als auch auf einem entfernten Rechner betrieben werden können. Ein Email soll durch entsprechende Markierung im Betreff-Feld und nochmaligem Absenden des bereits gesendeten Emails wieder zurückgezogen werden können. Dieses zweite Email zum Zurückziehen wird im Folgenden Recall-Email genannt.

Die Aufgabe des Programmes *SpontaneousRecall* ist, in der Mailbox nach Recall-Emails zu suchen, das ursprüngliche Email durch verschiedene Techniken zu finden und zu löschen bzw. als zurückgezogenes Email zu markieren. Auch hier soll sich der Absender durch eine Bestätigung über die Zurücknahme informieren lassen können.

3.2 Vorhandene Umgebung

Als Zielumgebung für die Programme wurden die beiden Mailserver mail.uibk.ac.at und mail2.uibk.ac.at der Universität Innsbruck definiert. Bei diesen Servern handelt es sich um Unix-Rechner mit folgenden für diese Arbeit relevanten Eigenschaften:

- Zugriff über SecureShell möglich
- Schreibrechte im Benutzerverzeichnis
- Java 2 Runtime Environment 1.5 installiert
- Aktivierung von Procmail und Konfiguration von .procmailrc möglich

SpontaneousRecall soll zusätzlich auch auf beliebigen Rechnern mit installierter Java Runtime grösser 1.5 verwendet werden können.

3.3 Eingesetzte Technologien

Um die Programme auf möglichst vielen Betriebssystemen betreiben zu können fiel die Entscheidung zugunsten der **Programmiersprache Java** [4] aus. Es gibt für sehr viele Betriebssysteme eine Java Runtime Umgebung und beim Grossteil ist eine solche bereits vorinstalliert. Die Schnittstelle zwischen Applikation und Mailserver wird durch die **JavaMail API** [5] gebildet. Diese API vereinheitlicht den Zugriff auf verschiedene Mailserver und verschiedene Typen von Emails und erleichtert dadurch das Abholen, Verarbeiten und Senden von Emails.

Für das Logging in verschiedenen Detailstufen kommt die Bibliothek **log4j** [6] der Apache Software Foundation zum Einsatz. Tests der diversen Hilfsprogramme werden über das Testframework **JUnit** [7] abgedeckt. Sowohl *Vacation* als auch *SpontaneousRecall* werden, sofern sie direkt auf dem Mailserver betrieben werden, von **Procmail** [8] beim Einlangen von Emails gestartet. Procmail ist ein Mail Delivery Agent (MDA) der zur serverseitigen Filterung von Email Nachrichten verwendet wird aber auch über die Fähigkeit verfügt externe Programme aufzurufen.

3.4 Struktur der Arbeit

3.4.1 Package at.nolf

Wie in Abbildung 3.1 zu sehen, ist der Quellcode der Arbeit in drei Java-Packages unterteilt. Das Package *at.nolf* enthält die ausführbaren Programme *at.nolf.Configure*, *at.nolf.SpontaneousRecall* und *at.nolf.Vacation*. Die Funktionen der beiden Programme *Vacation* und *SpontaneousRecall* wurden bereits in Kapitel 3.1.1 und 3.1.2 kurz beschrieben. *Configure* ist ein Tool, welches die Konfiguration der beiden Lösungen vereinfachen soll.

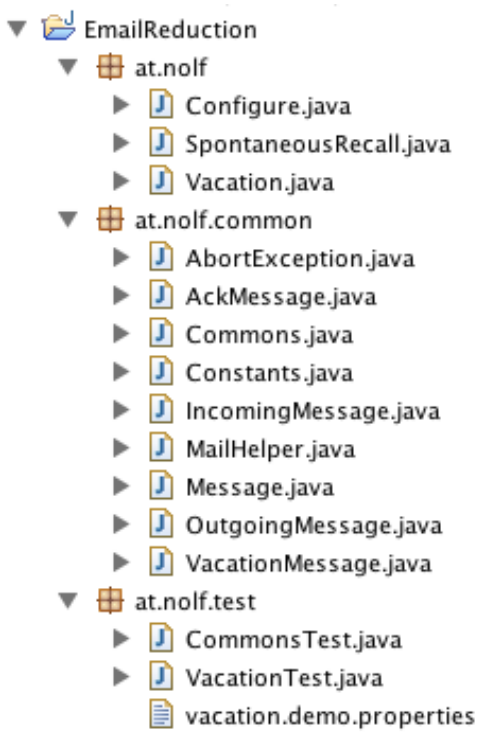

Abbildung 3.1: Gliederung des Quellcodes

Es stellt dem Benutzer einfache Fragen und befüllt die verschiedenen Konfigurationsdateien mit dessen Antworten.

3.4.2 Package at.nolf.common

Um Code-Redundanzen zu vermeiden wurden jene Funktionen und Daten in Klassen ausgelagert, die von den Hauptprogrammen gemeinsam gebraucht werden. Alle diese Klassen befinden sich im Package *at.nolf.common*.

Die Klasse *Message* und deren Subklassen (siehe Abbildung 3.2) bilden die verschiedenen Arten von Emails ab, die das System verarbeitet bzw. verschickt.

Die Klasse *AbortException* repräsentiert einen Fehler, der zum unweigerlichen Abbruch des jeweiligen Programmes führt. Diese Exception wird immer dann geworfen, wenn das weitere Ausführen des Programmes nicht mehr sinnvoll durchgeführt werden kann, sei es weil entsprechende Daten fehlen, ein Subsystem einen schweren Fehler meldet, der Mailserver nicht erreichbar ist, oder Ähnlichem.

Die Klasse *Constants* dient als zentrale Datensammlung für alle anderen Klassen. Sie enthält alle Konstanten wie Namen der Template-, Konfigurations- und Loggerdateien, Namen aller Keys in den Konfigurationsdateien, Standardwerte

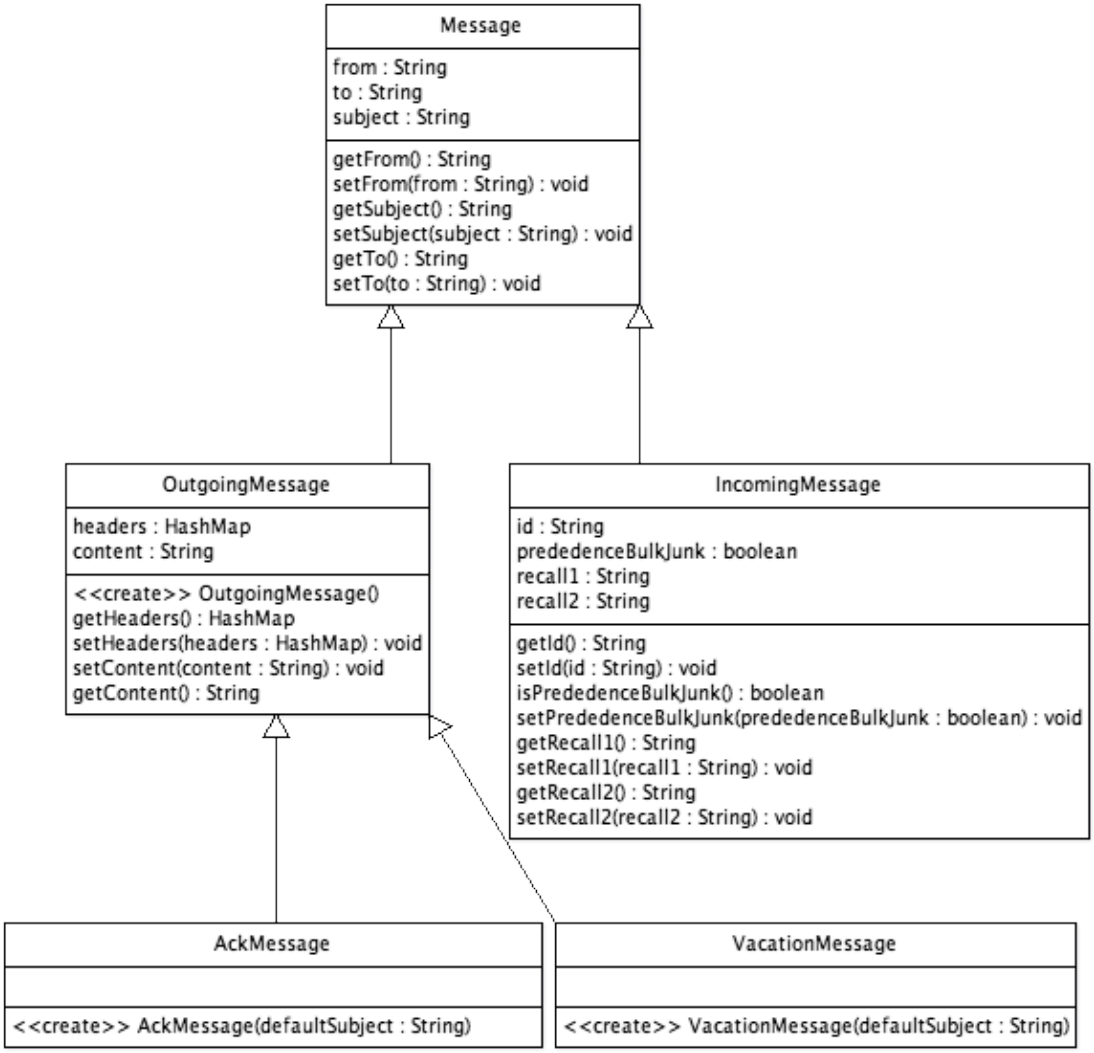

Abbildung 3.2: Klassen zur systeminternen Repräsentation von Emails

für Konfigurationen und Templates, Namen der Template-Platzhalter und vieles mehr.

Commons ist eine Helper-Klasse und bietet statische Funktionen unterschiedlicher Art an:

- Parsen von Emails, die als Text über den StandardInput an das Programm überreicht werden: dabei bietet *Commons* die Möglichkeit über die Methoden *matchesXXX()* abzufragen, ob die aktuelle Textzeile des Emails für das Programm relevante Informationen enthält, welche sodann mittels getXXX() extrahiert werden können. Um perfomante Vergleiche zu erzielen wird das Pattern-Matching von Java verwendet. Dafür werden die verschiedenen Suchen mittels regulären Ausdrücken als Pattern vorkompiliert um so auf jede Zeile rasch angewendet werden zu können.
- Hilfsmethoden für die Verwaltung der internen Datenbankdatei *vacation.db.properties* zur Erzeugung von neuen Einträgen und zum Finden bzw. Extrahieren von bestimmten Informationen.
- Methoden zum Ver- und Entschlüsseln des Passworts für den Mailaccount, welches in der Datei *connection.properties* verschlüsselt abgelegt wird.
- Funktion zum Erzeugen eines zufälligen Schlüssels konfigurierbarer Länge bestehend aus Zahlen und Buchstaben. Dieser Schlüssel wird in der Abwesenheitsnachricht mitgeschickt, auf die geantwortet werden kann.
- Methoden zum Laden und Speichern der verschiedenen Konfigurations-Dateien.
- Methoden zum Erzeugen und Senden von Bestätigungsnachrichten über das erfolgreiche oder nicht erfolgreiche Zurückziehen eines Emails.

MailHelper ist eine zentrale Hilfsklasse, die von anderen Programmen dazu verwendet wird eine Verbindung zum konfigurierten Mailserver aufzubauen bzw. zu beenden, Mails abzuholen, zu löschen und zu versenden. Ausserdem stellt die Klasse eine Methode *cloneMessage(Message, String, String)* zur Verfügung, die das übergebene Mail klont und den Betreff sowie die Headerinformationen erweitert. Damit können Emails auch als zurückgezogen gekennzeichnet werden.

3.4.3 Package at.nolf.test

Im Package *at.nolf.test* befinden sich JUnit-Tests, die allgemeine Funktionen und Hilfsmethoden testen, welche keine direkte Abhängigkeit zu einem Mailserver haben. Beispielsweise wird hier das Parsen eines per StandardInput übergebenen Emails und das Schreiben, Lesen und Aufräumen der Datenbankdatei *vacation.db.properties* getestet.

3.4.4 Weitere Dateien

Neben dem Jar-File und den darin enthaltenen Klassen sind auch verschiedene Template- und Konfigurations-Dateien von Nöten, deren Bedeutung hier kurz erläutert wird. Diese Dateien müssen teilweise von Hand erstellt bzw. angepasst werden, werden im Zuge der erstmaligen Konfiguration angelegt oder beim erstmaligen Lauf vom System selbst erzeugt.

Template-Dateien

Bei den so genannten Template-Dateien handelt es sich um Vorlagen für Emails, die vom System erstellt und versendet werden. Es gibt folgende Templates, welche von Hand an die jeweiligen Bedürfnisse angepasst werden müssen:

- *vacation.msg* Vorlage für die Abwesenheitsnachricht
- *ack_positive.msg* Vorlage für das Bestätigungsemail einer erfolgreichen Zurücknahme
- *ack_negative.msg* Vorlage für das Bestätigungsemail einer fehlgeschlagenen Zurücknahme

Detaillierte Informationen zum Aufbau und die Verwendung der Templates sind im Readme (Kapitel 6) zu finden.

Konfigurations-Dateien

Die Programme *Vacation* und *SpontaneousRecall* benötigen Informationen über den Mailaccount mit welchem sie arbeiten sollen. Diese Informationen umfassen den Hostnamen, Benutzernamen, Passwort, etc. und müssen in der Datei *connection.properties* hinterlegt werden. Das Verhalten von *SpontaneousRecall* kann man darüberhinaus mittels der Datei *recall.properties* genau spezifizieren. Dort kann konfiguriert werden ob zurückgezogene Emails endgültig gelöscht oder nur markiert werden sollen, welchen zusätzlichen Text ein Recall-Email im Betreff aufweisen muss und welche Algorithmen zum Auffinden des ursprünglichen Mails angewendet werden sollen. Nähere Ausführungen zum Thema Konfiguration finden sich im Readme (Kapitel 6).

Datenbank-Datei

Vacation verwaltet eine interne Datenbankdatei mit dem Namen *vacation.db.properties* um Informationen über bereits gesendete Abwesenheitsnachrichten persistent zu speichern. Diese Daten werden benötigt, um Abwesenheitsnachrichten nur in einem bestimmten Zeitabstand erneut an den gleichen Absender zu senden.

Darüberhinaus werden zu jedem Absender auch die eindeutige Message-ID der ursprünglichen Nachricht und der generierte, eindeutige Schlüssel der gesendeten Abwesenheitsnachricht gespeichert. Beide werden zum Auffinden und Identifizieren des ursprünglichen Emails im Falle eines Rückzuges benötigt.

Um eine grosse Datenansammlung zu verhindern, werden einmal täglich alle Einträge kontrolliert und abgelaufene entfernt.

Die Einträge der Datei sind Schlüssel-Wert-Tupel. Schlüssel ist die jeweilige Emailadresse des Absenders. Der Wert-Teil enthält mehrere Informationen, die durch den Separator \0 getrennt werden. Alle Einträge haben folgenden Aufbau:

```
AbsenderAdresse=MessageID\0Betreff\0Zufallsschluessel\0Zeitstempel
```

3.5 Programm Vacation

In diesem Kapitel wird das Programm *Vacation* und dessen Aufbau ausführlich beschrieben. Darüberhinaus können Einzelheiten dem Sourcecode bzw. der Klassendokumentation im Javadoc-Format (Html) entnommen werden.

3.5.1 Analyse des bestehenden Tools vacation.pl

Auf den Mailservern der Universität Innsbruck kommt für die Abwesenheitsbenachrichtigung das Perl Skript *vacation.pl* von Larry Wall zum Einsatz (Verzeichnis /usr/local/bin/). Das Skript sucht die Datei .vacation.msg und verwendet den darin enthaltenen Text als Vorlage für das Abwesenheitsmail, wobei alle Zeilen vor einer Leerzeile als Header-Felder interpretiert und die darauf folgenden Textzeilen als eigentlicher Email-Inhalt verwendet werden.

Nach Senden eines solchen Abwesenheitsemails werden die Adresse des Empfängers und der aktuelle Zeitstempel in einer Datei gespeichert, um bei Eintreffen eines weiteren Emails vom selben Absender innerhalb einer gewissen Zeitspanne eine erneute Abwesenheitsbenachrichtigung zu unterdrücken.

3.5.2 Umsetzung der Abwesenheitsbenachrichtigung in Vacation

Das Programm *Vacation* wird, wie in Kapitel 3.5.4 erläutert, durch eine Procmail-Anweisung gestartet, wobei das gerade eingetroffene Email als Text über den StandardInput dem Programm übergegeben wird. In der main-Methode von *Vacation* wird eine neue Instanz des Objekts erzeugt und im Konstruktor die beiden Dateien *vacation.db.properties* und *connection.properties* geladen. Anschließend wird die Methode *mopUpPropertyEntries()* ausgeführt, welche beim ersten Aufruf am jeweiligen Tag alle Einträge von *vacation.db.properties* kontrolliert und bereits abgelaufene Einträge löscht.

In der Methode *readAndProcessMessage()* wird das an das Programm übergebene Email über System.in gelesen, und es werden mittels der matchXXX()- und getXXX()-Methoden von *Commons* unter anderem folgende Informationen extrahiert:

- Message-ID
- From (Emailadresse des Absender)

- To (Emailadresse des Empfängers)
- Subject (Betreff des Emails)

Anschließend wird geprüft, ob die Absenderadresse bereits in der Datei *vacation.db.properties* vorhanden ist und wenn ja, ob die konfigurierte Zeitspanne bereits vergangen ist. Ist der Absender bekannt und die Zeitspanne noch nicht überschritten, so wird keine weitere Abwesenheitsnachricht verschickt und der Vorgang beendet. Andernfalls wird mit Hilfe der Methode *Commons.getRandomKey()* ein Zufallsschlüssel generiert und in der Methode *createVacationMessageContent()* eine Abwesenheitsnachricht erzeugt.

Zum Erzeugen der Nachricht wird die Templatedatei vacation.msg eingelesen, wobei alle Zeilen bis zur ersten Leerzeile als Header-Felder interpretiert werden und in die Header-Liste des Objekts *VacationMessage* kommen. Gesondert behandelt werden hier nur die Vorkommnisse von "From:"(Absender) und "Subject:"(Betreff) im Template. Diese Daten werden in die korrespondierenden Felder des Objekts *VacationMessage* übernommen.

Der Text nach der ersten Leerzeile wird auch zeilenweise ausgewertet und zur Variable "content" des Objekts *VacationMessage* hinzugefügt, wobei alle im Template definierten Platzhalter wie folgt ersetzt werden:

- $SUBJECT wird durch den Betreff des eingehenden Emails ersetzt
- $RECALL-LINE-1 wird durch die eigene Emailadresse umschlossen von "Recall"Tags ersetzt
- $RECALL-LINE-2 wird durch den generierten, eindeutigen Schlüssel umschlossen von "Recall"Tags ersetzt

Ein Beispiel einer fertigen Abwesenheitsnachricht zeigt sich in Abbildung 3.3. Die sogenannten Recall-Tags haben die Form <RECALL-1> </RECALL-1> und <RECALL-2> </RECALL-2>, wobei sich die jeweiligen Informationen zwischen den beiden Blöcken befinden. Diese Tags werden dazu benötigt die beiden Schlüssel durch reguläre Ausdrücke im Antwortmail wiederzufinden.

Wenn keine Templatedatei existiert, wird eine Standard-Vorlage verwendet, welche in der Klasse *Constants* definiert wurde. Nachdem das Objekt *VacationMessage* fertig befüllt ist, kann es mittels der Methode *sendMessage(Message)* des *MailHelper* verschickt werden. Abschliessend wird noch ein Eintrag mit den aktuellen Informationen in die Datenbankdatei *vacation.db.properties* geschrieben.

3.5.3 Umsetzung der Funktion zum Zurückziehen eines Emails

Wie schon in der Anforderung (Kapitel 3.1.1) beschrieben, wird das Zurückziehen eines Emails durch das Antworten auf ein Abwesenheits-Email ausgelöst, wobei sich der Absender über den Erfolg bzw. Misserfolg durch Angabe der

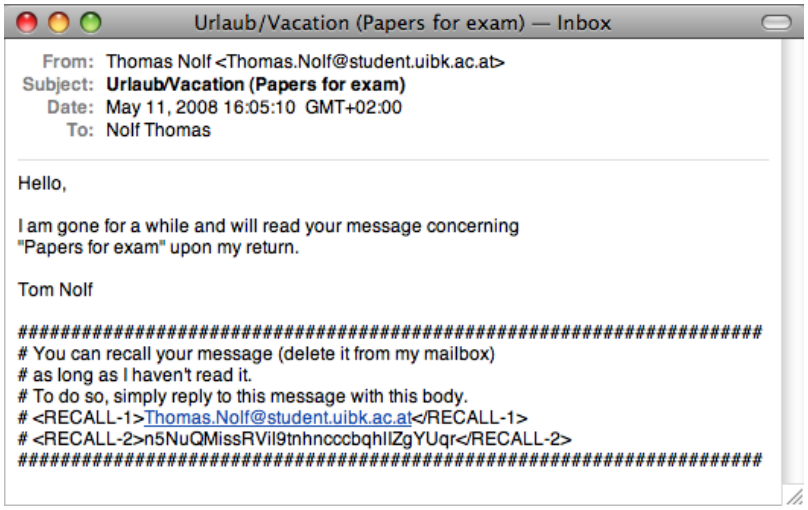

Abbildung 3.3: Beispiel einer fertigen Abwesenheitsnachricht

speziellen Markierung *ACK* zu Beginn des Betreffs informieren lassen kann.

Auch die Funktion zum Zurückziehen eines Emails befindet sich in der Klasse *Vacation* und wird durch Angabe des Parameters "recall" beim Start der Anwendung aktiviert. Der Start von *Vacation* und die Verarbeitung des ankommenden Recall-Emails verläuft analog zu dem in Kapitel 3.5.2 beschriebenen Ablauf, nur werden in der Methode *readAndProcessMessage()*, in der das ankommende Mail gelesen und verarbeitet wird, nun zusätzlich auch noch die Schlüssel der beiden Recall-Tags extrahiert.

Sind alle notwendigen Informationen vorhanden, so wird als erstes geprüft, ob die Absenderadresse des Recall-Emails in der Datenbankdatei vorhanden ist und der Absender somit auch wirklich eine Abwesenheitsnachricht bekommen hat. Ist dies der Fall, werden die Schlüssel des Recall-Emails mit den in der Datenbankdatei gespeicherten Schlüssel verglichen um die Authentizität des Recall-Emails zu prüfen. Stimmen alle überein, so wird die zugehörige Message-ID aus dem Eintrag der Datenbankdatei gelesen und der Methode *deleteByMessageId(String id)* des *MailHelper* übergeben. Diese Methode stellt eine Verbindung zum Mailserver her, sucht das zu löschende Mail anhand seiner Message-ID und löscht es.

Hat der Absender im Betreff des Recall-Emails noch die spezielle Markierung *ACK* angegeben, so wird nun das Handling zur Benachrichtigung gestartet. War das Auffinden und Löschen erfolgreich, so wird das Template *ack_positive.msg* eingelesen, andernfalls das Template *ack_negative.msg*. Diese beiden Vorlagen haben den selben Aufbau wie *vacation.msg* und werden auch entsprechend interpretiert und versendet.

3.5.4 Einbinden von Vacation

Wie im Readme (Kapitel 6) ausführlicher beschrieben, wird *Vacation* durch folgende Einträge in der Datei .procmailrc eingebunden:

```
# comment these 3 lines if you are not on vacation
:0ic
*
| java -cp EmailReduction.jar at.nolf.Vacation

:0B
* .*\<RECALL-1\>thomas.nolf@student.uibk.ac.at\</RECALL-1\>
| java -cp EmailReduction.jar at.nolf.Vacation recall
```

3.5.5 Beispielablauf

Das Sequenzdiagramm in Abbildung 3.4 stellt einen kompletten Ablauf von *Vacation* grafisch dar:

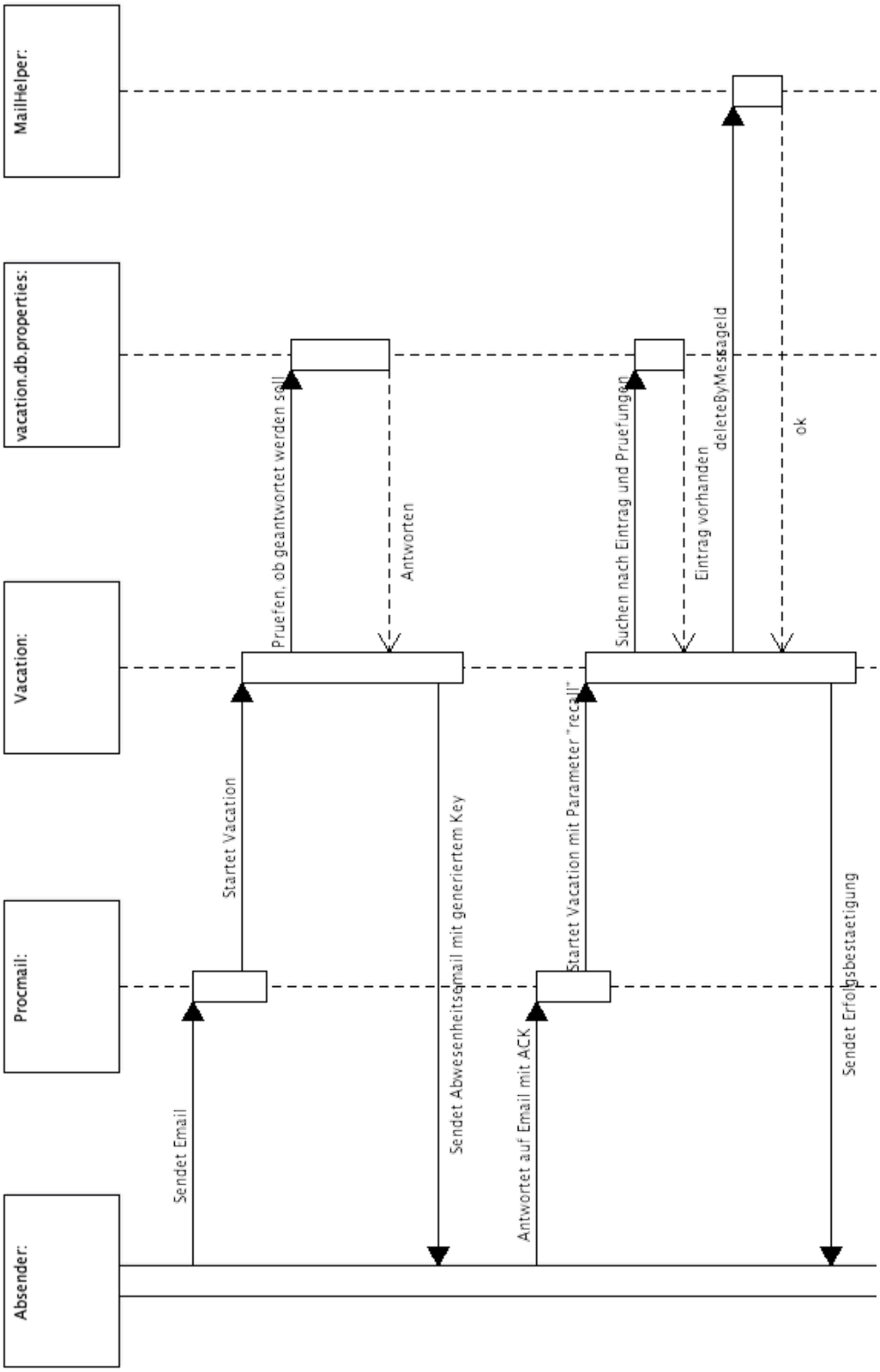

Abbildung 3.4: Erfolgreicher Ablauf von Vacation

3.6 Programm SpontaneousRecall

In diesem Kapitel wird das Programm *SpontaneousRecall* und dessen Aufbau beschrieben. Nähere Einzelheiten können dem Sourcecode bzw. der Klassendokumentation im javadoc-Format (Html) entnommen werden.

3.6.1 Ansätze und Anforderungen

Das Programm *SpontaneousRecall* muss nicht notwendigerweise am Mailserver laufen und wurde deshalb so entworfen, dass Recall-Emails nicht über den StandardInput übergeben werden, sondern das Programm die Recall-Emails selbständig im Postfach sucht. Wenn *SpontaneousRecall* doch am Mailserver betrieben wird, so gibt Procmail bei Ankunft von neuen Emails nur noch den Anstoss zum Laufen.

Im Gegensatz zu *Vacation* ist im Recall-Email kein eindeutiger Schlüssel vorhanden, der in Zusammenhang mit dem ursrprünglichen Mail steht und für dessen Auffinden verwendet werden kann. Stattdessen mussten andere Möglichkeiten gefunden werden um das Recall-Email mit dem ursprünglich gesendeten Email in Verbindung zu bringen. Dabei kristallisierten sich folgende Anforderungen an das System heraus:

- Das zurückzuziehende Mail muss möglichst zuverlässig und sicher gefunden werden.
- Wenn mehr als ein passendes Email zu einem Recall-Email gefunden wird, darf keines gelöscht werden.
- Das Recall-Email muss immer gelöscht werden, auch wenn kein Rückzug durchgeführt wurde oder ein Fehler auftrat.
- Emails sollen bei Rückzug gänzlich gelöscht oder auch nur markiert werden können.
- Das Zurückziehen von Emails soll bei allen gängigen Emailclients funktionieren.

3.6.2 Grober Ablauf

Ausgangssituation

SpontaneousRecall wurde so konfiguriert, dass ein Recall-Email die Markierung *RECALL-ME* bzw. *RECALL-ME-ACK* zu Beginn des Betreffs des weitergeleiteten Emails aufweisen muss (siehe Readme in Kapitel 6). Weiters sind sowohl das ursprüngliche Email als auch das dazugehörige, korrekt formatierte Recall-Email im Postfach vorhanden. In den Abbildungen 3.5 und 3.6 sind beispielhaft ein Email und dessen Recall-Email nach den Anforderungen der Ausgangssituation zu sehen.

Abbildung 3.5: Beispiel eines gesendeten Emails

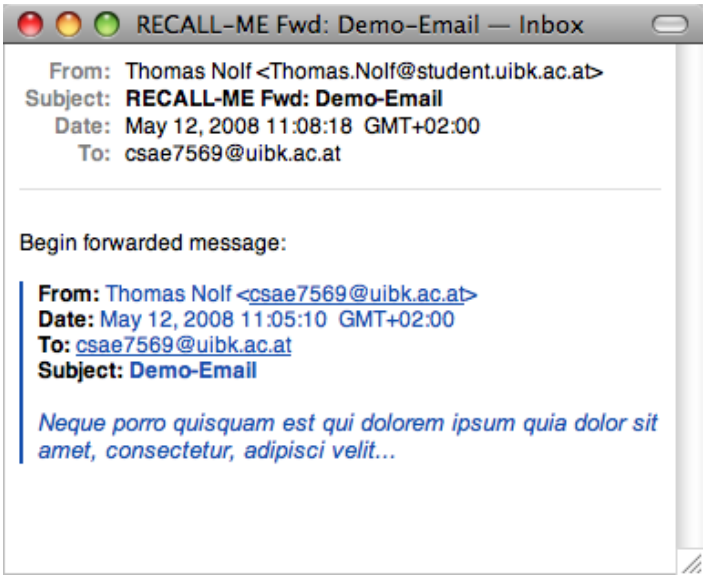

Abbildung 3.6: Beispiel eines Recall-Emails

Programmablauf

Nach Start der Anwendung wird in der Methode *init()* über den *MailHelper* eine Verbindung zum Postfach hergestellt und die Konfigurationsdatei *recall.properties* eingelesen. Im Anschluß daran wird die Hauptmethode *runSolo()* gestartet in welcher zu Beginn alle Emails gesucht werden, die im Betreff die konfigurierte Markierung (in diesem Beispiel: RECALL-ME) aufweisen. Anschließend wird für jedes gefundene Recall-Email das selbe Schema durchlaufen: es wird versucht das ursprüngliche Mail zu finden, indem nach der Message-ID gesucht wird (Details dazu im Kapitel 3.6.3). Wenn es gefunden wurde, wird es, je nach Konfiguration, gelöscht oder nur markiert (Details zur Markierung sind im Kapitel 3.6.5 zu finden).

Wird das ursprüngliche Email durch diesen Vorgang nicht gefunden, so werden alle vorhandenen Emails vom selben Absender durchsucht und geprüft, ob der Betreff des jeweiligen Emails im Betreff des Recall-Emails enthalten ist. Dadurch wird die Anzahl der Möglichkeiten bereits auf ein Minimum reduziert. Bei den noch übrigen Möglichkeiten wird geprüft, ob das jeweilige Email im Recall-Email enthalten ist (Details hierzu folgen im Kapitel 3.6.4).

Nur wenn exakt eine Übereinstimmung gefunden wurde, wird diese gelöscht oder markiert. Hat der Absender die Markierung RECALL-ME-ACK verwendet und dadurch eine Bestätigung angefordert, so wird äquivalent zur Vorgehensweise bei *Vacation* das Template *ack_positive.msg* oder *ack_negative.msg* geladen, ein Email erstellt und mittels *MailHelper* versendet.

3.6.3 Finden des Emails mittels Message-ID

Der Internet Message Format Standard (RFC2822) empfiehlt, dass jedes Email ein Feld Message-ID mit einer weltweit eindeutigen Kennung enthalten soll. Wird ein Email weitergeleitet, so erhält es eine neue eindeutige Kennung und die Message-ID der ursprünglichen Nachricht soll ins Feld *References* übernommen werden. Nachdem das Recall-Email im Feld *References* also schon die Message-ID des zu löschenden Emails enthält, gestaltet sich dessen Auffinden als sehr einfach und sicher.

Das einzige Problem bei dieser Lösung ist, dass diese Felder nicht verpflichtend verwendet werden müssen und im Standard nur empfohlen werden. Tests mit verschiedenen Mailclients haben ergeben, dass das Feld Message-ID immer mit einer eindeutigen Kennung belegt war, jedoch dieser Wert nur von wenigen Clients auch ins Feld *References* übernommen wurde.

3.6.4 Finden des Emails mittels Textvergleich

Enthält das Recall-Email kein Feld *References*, so wird als Ausweichlösung ein Vergleich der Mailinhalte durchgeführt. Aufgrund von folgenden Fakten können zwei Emails jedoch nicht 1:1 miteinander verglichen werden:

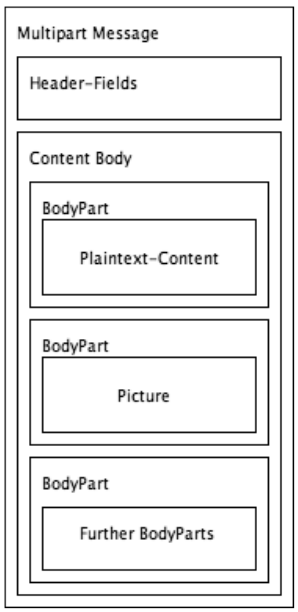

Abbildung 3.7: Aufbau einer Multipart-Message

- Es gibt unterschiedliche Arten von Emails (Plaintext, Html, Html mit Plaintext-Alternative, Multipart, Emails mit Anhang, etc.).
- Mailclients zitieren beim Weiterleiten eines Plaintext-Emails den Inhalt der ursprünglichen Nachricht unterschiedlich.
- Der Inhalt von Html-Emails wird beim Weiterleiten teilweise stark verändert wodurch ein Vergleich nicht möglich ist.
- Teilweise werden Anhänge beim Weiterleiten von Emails nicht automatisch mitgeschickt.
- Manche Emailclients wie Mozilla Thunderbird extrahieren den Inhalt des ursprünglichen Mails beim Weiterleiten und senden ihn gesondert als "Nested-Part"mit.

Neben Emails mit reinem Textinhalt werden die so genannten Multipart-Messages am häufigsten verwendet. Diese Art von Email kann beliebigen Inhalt enthalten, wobei die Teile auch verschachtelt sein können (siehe Abbildung 3.7). Multipart-Messages werden beispielsweise auch dazu verwendet um bei Html-Emails eine Plaintext-Alternative mitzusenden oder Bilder in den Inhalt einzubetten. Wie in Abbildung 3.8 zu sehen ist, wird diese Struktur von der Java Mail API mittels der Klasse *Multipart* abgebildet.

Das Programm *SpontaneousRecall* vergleicht zum Auffinden des zu löschenden Emails nur die Plaintext-Teile, wobei hier jede beliebig verschachtelte Struktur berücksichtigt wird. Dazu wurde die Methode *getAllTextParts(Part, Collection)* entwickelt, die rekursiv alle Emailteile traversiert, jegliche Text-Teile

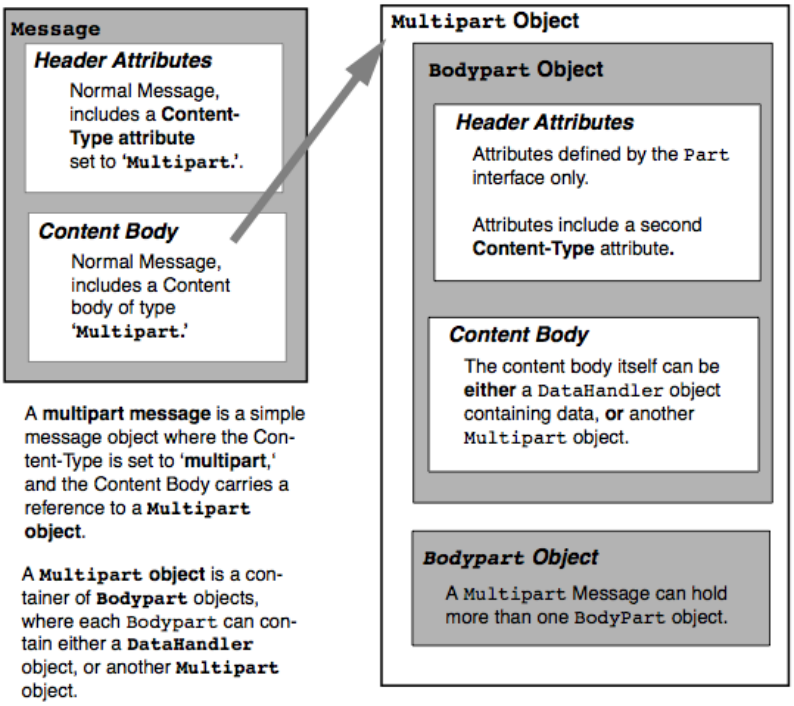

Abbildung 3.8: Multipart-Message der Java Mail API

des Emails findet und in einer Liste sammelt. Hier ist eine gekürzte Fassung der Methode:

```
void getAllTextParts(Part p, Collection<String> list) {
    if (p.isMimeType("text/plain")) {
        list.add((String)p.getContent());
    } else if (p.isMimeType("text/html")) {
        // ignore
    } else if (p.isMimeType("multipart/*")) {
        Multipart mp = (Multipart)p.getContent();
        for (int i = 0; i < mp.getCount(); i++)
                getAllTextParts(mp.getBodyPart(i), list);
    } else if (p.isMimeType("message/rfc822")) {
        // nested message
        getAllTextParts((Part)p.getContent(), list);
    }
```

Anzumerken ist hier, dass durch die Verwendung der Methode *isMimeType(String)* nur der Inhaltstyp des jeweiligen Teils abgefragt wird und nicht der ganze Teil erst vom Server geladen werden muss. Erst wenn mit *p.getContent()* auf ein Element zugegriffen wird, wird es geladen.

Die Methode *getAllTextParts(Part, Collection)* wird zu Beginn für das Recall-Email und dann in einer Schleife für alle übrigen Emails ausgeführt, deren

Absender und Betreff zum Recall-Email passen. Dabei wird in jedem Schleifendurchlauf nach Sammeln der Text-Teile eines Emails mittels der Methode *testMatchParts(List, List)* ein Vergleich mit dem Recall-Email durchgeführt.

```
1:int testMatchParts(List origParts, List recallParts) {
2:  int numberOrigParts = origParts.size();
3:  // represents number of matching characters of this message
4:  int rankingPoints = 0;
5:
6:  for (String orig : origParts) {
7:    for (String recall : recallParts) {
8:      if(recall != "") {
10:      if(recall.contains(orig)) {
11:        // match!
12:        numberOrigParts--;
13:        rankingPoints += orig.length();
14:        recall = ""; // do use each part only once!
15:        break; // next orig part
16:      } // if
17:    } // if
18:  } // for
19:
20:  if(numberOrigParts <= 0) {
21:    return rankingPoints;
22:  } else {
23:    return 0;
24:  }
```

Diese Methode sucht für jeden Teil des zu untersuchenden Emails einen entsprechenden Teil im Recall-Email (Zeilen 6 bis 18), wobei jeweils geprüft wird, ob der Teil in einem Teil des Recall-Emails enthalten ist (Zeile 10). Dies passiert, weil der Inhalt des Recall-Emails durch die Weiterleitung meist zitiert ist und dadurch zu Beginn jeder Zeile zusätzliche Zeichen enthalten kann.

Wenn zwei passende Teile gefunden wurden, wird der Zähler der noch zu prüfenden Teile vermindert (Zeile 12), die Anzahl der übereinstimmenden Zeichen zur Variable *rankingPoints* addiert (Zeile 13) und der Teil des Recall-Emails gelöscht, damit dieser nicht für weitere Prüfungen verwendet wird (Zeile 14).

Nur wenn für jeden Teil des zu untersuchenden Emails eine Entsprechung im Recall-Email gefunden wurde, wird die Anzahl der RankingPoints retourniert. Andernfalls wird die Zahl 0 zurückgegeben, was auf einen negativen Vergleich hindeutet (Zeilen 20 bis 24).

Schlussendlich ist das zu löschende Email jenes, welches am Ende aller Vergleiche als einziges die maximale Anzahl an RankingPoints bekommen hat.

3.6.5 Markieren von Emails

Es kann per Konfiguration festgelegt werden, dass zurückgezogene Emails nicht gelöscht, sondern nur als solche markiert werden (siehe Readme im Kapitel 6). Nachdem Emails am Mailserver nicht mehr modifiziert werden können, musste hier eine andere Möglichkeit gefunden werden. Ziel war es das Email exakt zu duplizieren, das Betreff-Feld zu erweitern und ein zusätzliches Header-Feld zu befüllen.

Dazu wurde die Methode *cloneMessage(Message, String, String)* im *MailHelper* implementiert, welche ein Email des Typs *javax.mail.Message*, den Prefix für den Betreff und den Namen des Header-Feldes entgegen nimmt. Anhand der übergebenen Message wird eine neue *MimeMessage* erzeugt, wobei deren Betreff um den Prefix erweitert und das neue Header-Feld hinzugefügt wird.

Dieser "erweiterte Klon" kann nun wiederum über die Klasse *MailHelper* an die eigene Emailadresse versendet werden und stellt im Postfach das zurückgezogene Email dar.

3.6.6 Einbinden und Verwenden von SpontaneousRecall

Wenn *SpontaneousRecall* direkt am Mailserver betrieben wird, kann es durch folgende Einträge in der Datei .procmailrc bei Eintreffen von Recall-Emails automatisiert gestartet werden:

```
:0wc
| $DELIVER +INBOX

:0Wi
* ^Subject:.*RECALL-ME.*
| java -cp EmailReduction.jar at.nolf.SpontaneousRecall

# E
:0
|
```

Zu beachten ist, dass *SpontaneousRecall* erst gestartet werden darf, wenn das Recall-Email bereits ins Postfach zugestellt wurde (Zeile 2). Eine ausführliche Erklärung der Zeilen ist im Readme (Kapitel 6) zu finden.

Das Programm kann aber auch auf einem entfernten Rechner gestartet werden, indem einer der folgenden Befehle in der Kommandozeile ausgeführt wird:

```
java -cp EmailReduction.jar at.nolf.SpontaneousRecall
```

Oder wenn das Programm als Dämon in bestimmten Zeitintervallen laufen soll:

```
java -cp EmailReduction.jar at.nolf.SpontaneousRecall daemon
```

Das Zeitintervall des Dämons kann in der Konfigurationsdatei *recall.properties* festgelegt werden und ist standardmässig auf fünf Minuten eingestellt.

3.6.7 Beispielablauf

Das Sequenzdiagramm in Abbildung 3.9 stellt einen kompletten Ablauf von *SpontaneousRecall* grafisch dar:

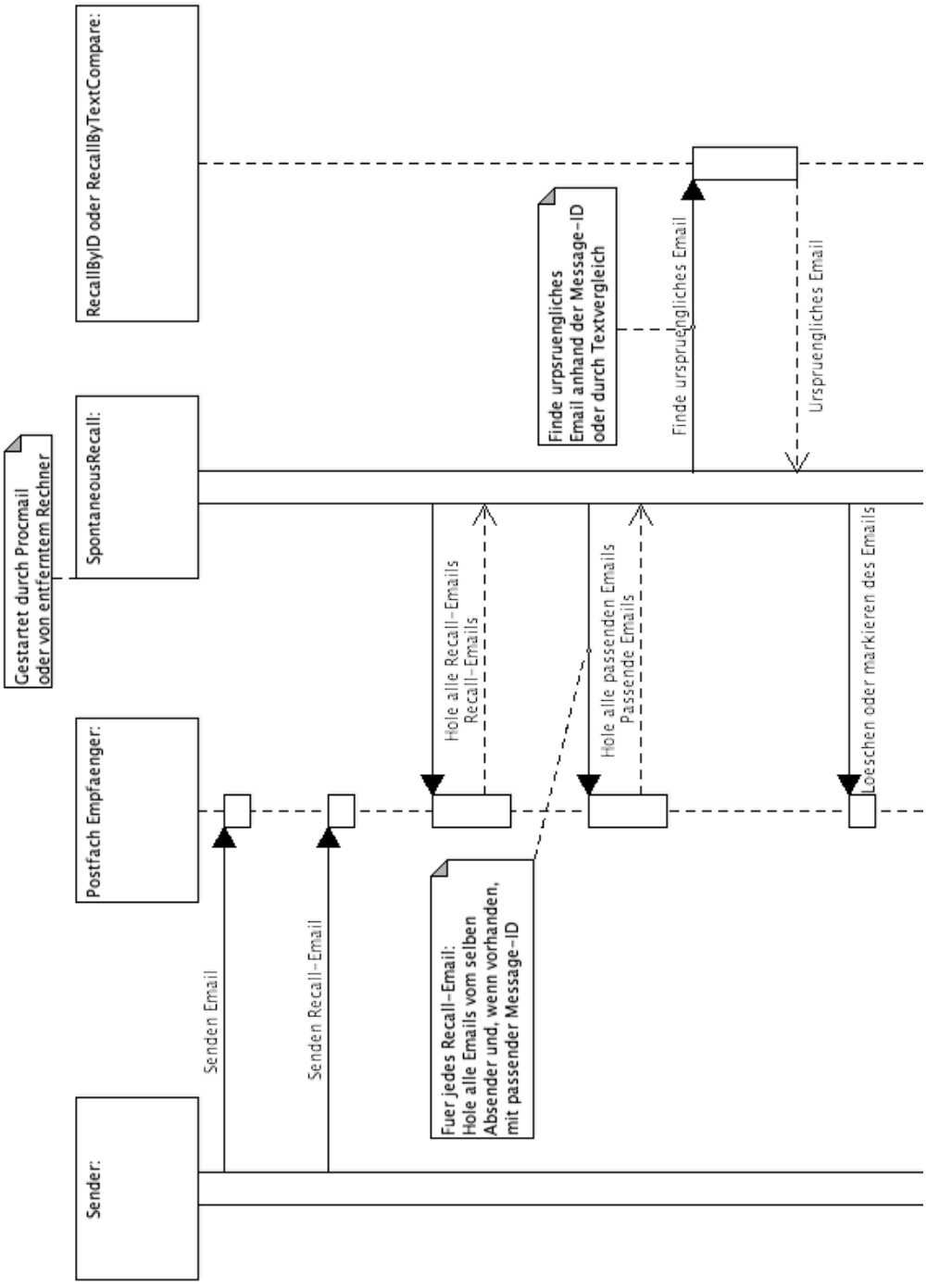

Abbildung 3.9: Erfolgreicher Ablauf von SpontaneousRecall

4 Test der Lösung

Die Tests der Lösungen lassen sich grob in drei Teile untergliedern, wobei der erste Teil programmatisch und die beiden Programme *Vacation* und *SpontaneousRecall* manuell getestet wurden. Bei den manuellen Tests wurde darauf geachtet, dass möglichst viele verschiedene Konstellationen, Konfigurationen und die gängigsten Emailclients getestet wurden. Dabei kamen folgende Emailclients zum Einsatz:

- Apple Mail
- Mozilla Thunderbird
- Ximian Evolution
- Microsoft Outlook
- Microsoft Outlook Express

4.0.8 Test der Klasse Commons

Die Funktionalität der Klasse *Commons* wurde durch JUnit-Tests sichergestellt. Dabei wurden folgende Punkte besonders beachtet:

- Umgang mit Konfigurationsdateien (finden, lesen, schreiben, aufräumen)
- Parsen und Filtern eines Emails, welches als Text an den StandardInput übergeben wird
- Ver- und entschlüsseln des Passworts und konvertieren in hexadezimale Darstellung

4.0.9 Test von Vacation

Bei diesen Tests lag der Schwerpunkt sowohl bei der Verarbeitung des Templates, als auch beim Rückziehen von Emails. Dabei wurde ohne Benutzertemplate, mit unvollständigen Templates, verschiedenen Headers, usw. getestet.

Der Vorgang des Rückziehens wurde von den weiter oben angeführten Emailclients durchgeführt, wobei nicht nur der Normalfall, sondern auch diverse Fehlerfälle getestet wurden (Bsp.: Antworten auf eine Abwesenheitsnachricht von einem anderem Mailaccount, falscher Schlüssel, etc.).

Dabei wurden bei jedem Testdurchlauf auch die Logdateien, Statistikdateien und das Postfach auf korrekten Inhalt und Abnormalitäten überprüft.

4.0.10 Test von SpontaneousRecall

Auch die Tests von *SpontaneousRecall* lassen sich grob in zwei Schwerpunkte unterteilen. Zum Einen wurde das Zurückziehen von Emails anhand der Message-ID und Referers getestet und zum Anderen die Variante mit dem Textvergleich, wobei für die Zweitere durch die verschiedenen Emailarten und Emailclients wesentlich mehr Testfälle erforderlich waren.

Auch hier wurden nicht nur die Normalfälle getestet, sondern Spezialfälle wie zum Beispiel:

- Senden des Recall-Emails von einem anderen Mailaccount als jenem, von dem das originale Email versendet wurde
- Erzeugen von zwei Emails mit identischem Inhalt und Löschen eines dieser beiden (vor allem interessant für den Textvergleich)
- Zurückziehen eines Emails nach längerer Emailkorrespondenz (Test des Ranking-Systems, weil beim Zurückziehen mehrere Emails eine textuelle Übereinstimmung aufweisen)
- Testen des Fallbacks auf Textvergleich, wenn kein Referers-Feld vorhanden ist

Bei den Tests des Textvergleichs wurden verschiedenste Emails mit unterschiedlichem Inhalt von den verschiedenen Emailclients gesendet. Dabei wurden Emails mit Text- oder Html-Inhalt, mit Anhängen und ohne, mit verschachtelten Inhaltstypen und vieles zudem noch kombiniert verwendet.

Außerdem wurden alle Testfälle sowohl direkt auf dem Mailserver als auch auf einem entfernten Rechner durchgeführt und das Löschen sowie das Markieren von zurückgezogenen Emails getestet.

5 Zusammenfassung

5.1 Allgemeines

Folgende Software wurde bei dieser Arbeit verwendet:

- Eclipse 3.2.2 für Mac OS X
- Fat Jar Eclipse Plugin 0.0.27 (Zum Erzeugen der ausführbaren Jar-Datei)
- TeXShop 2.10 (Zum Erstellen dieses Dokuments)

Folgende Ordnerstruktur befindet sich auf der Programm-CD:

- *source* (Gesamtes Eclipse-Projekt mit Quellcode)
- *target* (Ausführbare Jar-Datei und Konfigurationsdateien)
- *documents* (Dokumentation der Arbeit im Latex- und PDF-Format, sowie Javadoc des Quellcodes im Html-Format)

5.2 Rückblick und Danksagung

In dieser Arbeit wurden zwei Programme implementiert, die zur benutzerunterstützten Reduktion von Emails beitragen sollen. Zum Einen können Benutzer dadurch Emails zurückziehen, die aufgrund der Abwesenheit des Empfängers nicht benötigt werden und zum Anderen können beliebige Emails zurückgenommen werden, die sich nachträglich als nicht notwendig herausstellen.

Dazu wurde zu Beginn der Arbeit die vorhandene Umgebung auf den Mailservern der Universität Innsbruck analysiert. Es folgte eine Einarbeitungsphase in die verschiedenen Tools und APIs. Anschließend wurde das Design festgelegt und die Implementierung vorgenommen. Den Abschluß bildeten Testreihen und die Erstellung dieses Dokuments.

Abschließend möchte ich noch meinem Betreuer Dr. Michael Welzl danken, der die Idee für diese nützliche Arbeit hatte und mich dabei sehr unterstützte. Vielen Dank!

Literaturverzeichnis

[1] Microsoft Outlook:
http://office.microsoft.com/en-us/outlook/HA010917601033.aspx
(Letzter Zugriff: 06.05.2008)

[2] BigString: *http://www.bigstring.com* (Letzter Zugriff: 09.05.2008)

[3] YankBack: *http://www.yankback.com* (Letzter Zugriff: 11.05.2008)

[4] Java: *http://java.sun.com/javase/* (Letzter Zugriff: 11.05.2008)

[5] JavaMail API: *http://java.sun.com/products/javamail/*
(Letzter Zugriff: 25.04.2008)

[6] Apache log4j: *http://logging.apache.org/log4j/* (Letzter Zugriff: 22.03.2008)

[7] JUnit: *http://www.junit.org* (Letzter Zugriff: 14.02.2008)

[8] Procmail: *http://procmail.org und*
http://www.uibk.ac.at/zid/systeme/mail/procmail/index.html
(Letzter Zugriff: 19.03.2008)

6 Anhang: Readme

Email Reduction through SpontaneousRecall & Vacation

Version 1.0
Release date: 2008-04-17

by Tom Nolf

University of Innsbruck
thomas.nolf@student.uibk.ac.at

1 Introduction

You just downloaded a Java package that was developed to give people the possibility to recall emails that they already sent to you. Although there is only one jar-file there are two independently working applications, namely **Vacation** and **SpontaneousRecall**.

1.1 Vacation

at.nolf.Vacation was designed to run directly on a mailserver. As its name suggests, this program is capable of responding to emails with vacation messages. *Vacation* includes a random key in the vacation message so that people can simply reply to it in order to recall their initial email. *Vacation* uses an email template to create vacation messages and does only respond with a vacation message to the same sender within a configurable time interval.

1.2 SpontaneousRecall

at.nolf.SpontaneousRecall enables users to recall any email as long as it hasn't been already fetched. People can resend a sent email marking it with a special label in the subject, which is then called a recall email. When receiving such a recall email, *SpontaneousRecall* tries to find the initial appropriate email by applying various algorithms. Once it has found exactly one match the email gets either deleted or marked as recalled to be filtered by rules of various email clients. The behaviour is configurable.

SpontaneousRecall can operate in three different ways. It can be instantiated and run directly on the mailserver, started manually on a remote machine to run once, or run as a daemon on a remote machine in a configurable time interval.

Application	Utilization	Runs on
Vacation	Recall emails by replying to a vacation message	Mailserver
SpontaneousRecall	Recall any email	Mailserver or Remote Machine

Table 1. Overview of Utilization

2 Installing, Configuring and Using it

2.1 Vacation

Example Sequence for Understanding
After someone sends you an email (s)he gets an automatic vacation message where you may suggest the possibility to recall his email. In order to recall the initial email the person simply has to reply to the vacation email. If (s)he additionally wants to get an acknowledgement, (s)he has to insert the label *ACK* at the very beginning of the subject when replying to the vacation email.

Requirements

- Write access to mailserver by ftp, scp or similar to upload the files
- Installed JRE 5 or higher on the mailserver
- Installed and running procmail

Installation
Installation is very easy. Just upload the jar-file into the root directory of your mailserver and edit .procmailrc with your favorite editor (ie. vim) to bring it to instantiate *Vacation* upon arrival of new email. This can be achieved by copying the following snippet in your .procmailrc before the part where the email gets delivered to INBOX:

```
# comment those 3 lines if you are not on vacation
:0ic
*
| java -cp EmailReduction.jar at.nolf.Vacation

:0B
* .*\<RECALL-1\>thomas.nolf@student.uibk.ac.at\</RECALL-1\>
| java -cp EmailReduction.jar at.nolf.Vacation recall
```

The first three lines after the comment call *Vacation* upon each arrival of email so that it can send a vacation message if necessary. You should comment those 3 lines with the character '#' if you are not on vacation.

The second block is responsible for recognizing a reply to a vacation message and starting *Vacation* in recall mode. You should put your email address between the tags, which you specified in the configuration file connection.properties (key: mail.smtp.from, see chapter 2.7 for details).

Template for Vacation message
Since *Vacation* uses a template to create a vacation message, one can create and modify it ad libitum. The template must be a text file named vacation.msg which has to be located in the same directory as the jar-file. Here is the content of my vacation.msg:

```
From: Thomas Nolf <thomas.nolf@student.uibk.ac.at>
Subject: Urlaub/vacation (Re: $SUBJECT)
Precedence: bulk

I am on vacation until xxxx.
Please refer all urgent business to xxx@uibk.ac.at
Tom Nolf

######################################################################
# You can recall your message (delete it from my mailbox)
# as long as I haven't read it.
# To do so, simply reply to this message without altering the body.
# $RECALL-LINE-1
# $RECALL-LINE-2
######################################################################
```

Explanation: It is important to know that all lines until the first empty line are interpreted as header fields for the vacation message. You can add any header field according to the specification (RFC 2822).

$SUBJECT and the lines $RECALL-LINE-X are placeholders that are substituted by the application. $SUBJECT is replaced by the subject of the initial email, $RECALL-LINE-1 is substituted with the generated random key that makes it possible to delete the initial email by simply replying to it and $RECALL-LINE-2 will contain your email address to uniquely assign the message. Both $RECALL-LINE-1 and $RECALL-LINE-2 are mandatory in order to allow recalling of emails.

Template for Confirmation messages
Vacation (and also *SpontaneousRecall*) uses templates to create the confirmation messages about the successful or unsuccessful recall. Both templates must be text files in the same format as vacation.msg and named ack_positive.msg and ack_negative.msg respectively. One can use the placeholder $SUBJECT in these templates which will be replaced with the subject of the recall email. Here is the content of my ack_negative.msg:

```
Precedence: bulk
X-No-Archive: yes
Subject: Recall of $SUBJECT was not successful

Your recall of the message with the subject $SUBJECT was not
successful. Most likely the email has been fetched already.
```

Configuration
Vacation needs to know a few things about your mailaccount to work accordingly and expects this information in the file connection.properties. This file can easily be created using the tool *at.nolf.Configuration* or manually. See chapter 2.5 about how to create a configuration with this tool and chapter 2.7 to see all possible configurations.

Logs and Statistics
Detailed logs of *Vacation* are written to the file recall.log, but it also writes less detailed statistical information to a separate file called vacation_statistics.log to provide a better overview of the number of successful and unsuccessful recalls.

See chapter 2.6 for instructions on how to modify the loglevel, maximum size of logfiles, rotations, etc. if needed.

2.2 SpontaneousRecall

Example Sequence for Understanding
Imagine someone sends you an email and wants that to be undone as long as you haven't fetched it. In this case the person has to forward the sent message without altering its content and indicate it as a recall email by specifying a special recall label (i.e. *RECALL-ME*) at the very beginning of the subject. If (s)he additionally wants to get a confirmation of the recall, (s)he has to append the label *-ACK* to the defined recall label when forwarding the email.

Example:

```
Subject of initial email: Hello World
Subject of recall email: RECALL-ME Hello World
Subject of recall email with confirmation: RECALL-ME-ACK Hello World
```

General Configuration
SpontaneousRecall needs to know a few things about the mailaccount that it should handle. It shares this information with *Vacation* and therefore reads all needed configurations from the file connection.properties. Depending on whether *SpontaneousRecall* runs directly on the mailserver or on a remote machine, the hostnames may slightly differ (ie. localhost, mail2.uibk.ac.at).

Moreover the behaviour of *SpontaneousRecall* needs to be configured by providing a file named recall.properties. There you can define if a recalled email gets deleted or just marked, the label that should be specified in the subject to recall an email, and the algorithms that should be applied to find the initial email to recall.

Both configuration files can easily be created by using the tool *at.nolf.Configuration* or even manually. See chapter 2.5 about how to create a configuration with this tool and chapter 2.7 to see all possible configurations.

Templates for Confirmation Messages
See chapter 2.1 (Templates for confirmation messages) for details.

Logs and Statistics
Detailed logs of *SpontaneousRecall* are written to the file recall.log, but it also writes less detailed statistical informations to a separate file recall_statistics.log to provide a better overview of the number of successful and unsuccessful recalls.

See chapter 2.6 for instructions on how to modify the loglevel, maximum size of logfiles, rotations, etc. if needed.

2.3 SpontaneousRecall on Mailserver

Requirements

- Write access to mailserver by ftp, scp or similar to upload the files
- Installed JRE 5 or higher on the mailserver
- Installed and running procmail

Installation
Installation is very easy. If you didn't already upload the files for *Vacation*, do it now. Afterwards edit .procmailrc with your favorite editor (ie. vim) to bring it to instantiate *SpontaneousRecall* on arrival of email. This can be achieved by copying the following snippet to your .procmailrc **after** the part where the email gets delivered to INBOX.

It's very important that *SpontaneousRecall* is run *after* the email was delivered to your mailbox, because it searches and deletes the email directly from your mailbox and therefore the email has to be in your INBOX before running.

```
:0wc
| $DELIVER +INBOX

:0Wi
* ^Subject:.*RECALL-ME.*
| java -cp EmailReduction.jar at.nolf.SpontaneousRecall

# E
:0
|
```

The first block delivers the email to your INBOX. Pay attention to append the character 'c' also to your recipe so that the incoming email gets cloned before it gets delivered to your mailbox so that the processing of the procmailrc-file is continued.

The second block calls *SpontaneousRecall* to recall the email only if the recall email contains the label RECALL-ME in the subject. Be sure to use the same label that you configured in the file recall.properties.

The last block deletes a remaining email if it wasn't a recall email. This is necessary since procmail expects all emails to be worked up after processing .procmailrc. Since we cloned the email for delivering we may still have one email left if it wasn't handled in the block before.

2.4 SpontaneousRecall on a Remote Machine

Requirements

- Write access to an arbitrary directory
- Installed JRE 5 or higher

Installation and Configuration
Installation is done like on the mailserver. Just copy the files to an arbitrary directory on your machine. Run the tool *at.nolf.Configuration* or adjust the properties connection.properties and recall.properties by hand. See chapter 2.5 and 2.7 for details about the tool and the possibilities of configuration.

SpontaneousRecall can be started to run once by executing the following command in your command line:

```
java -cp EmailReduction.jar at.nolf.SpontaneousRecall
```

Or you may start it to run as a daemon by executing:

```
java -cp EmailReduction.jar at.nolf.SpontaneousRecall daemon
```

Note that the time interval of the daemon is configured in the file recall.properties.

2.5 Configuration Tool

To ease the process of configuration I developed a little tool that helps you with this. It's called *at.nolf.Configure* and can be run by executing the following command in your command line:

```
java -cp EmailReduction.jar at.nolf.Configure
```

It asks you some simple questions and writes your answers into appropriate configuration files. If there already exist some configuration files, *Configuration* tells you the current value so that you can keep the old value by simply hitting the enter key.

Sample question:

```
Authentication necessary? [ true | false ] (current: true):
Username: (current: csae7569):
```

2.6 Changing the Log Configuration

One can modify the loglevel, maximum size of logfiles, rotations, etc. by providing one's own log4j configuration. Just unzip the file EmailReduction.jar and copy the file log4j.properties to the root directory. Use the file as a template and change desired lines. The default log4j configuration is overwritten by your own configuration by setting a system property containing the path to your file. Example:

```
java -Dlog4j.configuration=file:./mylog4j.properties \
          -cp EmailReduction.jar at.nolf.SpontaneousRecall
```

2.7 Complete Summary of Configuration

Table 2. Configuration with connection.properties (*Vacation* and *SpontaneousRecall*)

Key	Description	Default
mail.pop3.host	Hostname of pop3 server	mail2.uibk.ac.at
mail.pop3.folder	Pop3 folder on server	INBOX
mail.pop3.auth	Is authentication at pop3 server necessary?	true
mail.transport.protocol	Protocol that is used to send an email	smtp
mail.smtp.host	Hostname of smtp server	localhost
mail.smtp.port	Port for smtp service	25
mail.smtp.from	Address of the sender (your email address)	You@uibk.ac.at
mail.smtp.starttls.enable	Allows communication over an encrypted layer	true
mail.smtp.auth	Is authentication at smtp server necessary?	false
session.debug	Log the process of sending an email very detailed	false
mail.user	Username for mailaccount (for both pop3 and smtp)	Your Username
mail.password	Encrypted password for mailaccount (pop3 and smtp) Has to be set by using the tool *at.nolf.Configure*	Your Password
vacation.timespan SendVacationMsgInMs	Time interval between sending two vacation messages to one and the same recipient (in ms)	604800000 (=1 week)

Table 3. Configuration with recall.properties (only for *SpontaneousRecall*)

Key	Description	Default
recall.indicatorRecallMsg	Text that has to be specified at the very beginning of the subject to indicate a recall message	RECALL-ME
recall.markMessageAsRecalled	If true, the email to recall is only marked as recalled and not deleted	true
recall.markMessageAsRecalled .subject	This prefix is added to the subject to indicate that this email has been recalled	RECALLED:
recall.markMessageAsRecalled .header	This header is added to the recalled email	Recalled
recall.recallByMsgId	If true, *SpontaneousRecall* will try to find the initial email to recall by means of the message-id	true
recall.recallByTextCompare	If true, *SpontaneousRecall* will try to find the initial email to recall by comparing the content	true
recall.respectDateLastRun	If true, *SpontaneousRecall* will only search for recall messages that arrived since the last run	true
recall.timespanDaemonRun	Time interval between two runs of *SpontaneousRecall* when it operates in daemon mode (in ms)	30000