



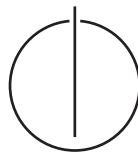
FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Bachelorarbeit in Informatik

**Myriad – a mailmerge tool for massive
parallel, yet individual email conversations**

Ludwig Schubert





FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Bachelorarbeit in Informatik

Myriad – a mailmerge tool for massive parallel, yet
individual email conversations

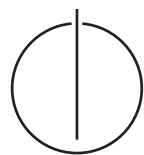
Myriad – ein Serienbrief Email-Tool für hochgradig
parallele, jedoch individualisierte Emailkonversationen

Author: Ludwig Schubert

Supervisor: Prof. Dr. Johann Schlichter

Advisor: Dr. Wolfgang Wörndl

Date: September 30, 2013



Ich versichere, dass ich diese Bachelorarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 30. September 2013

Ludwig Schubert

Abstract

This thesis introduces the myriad system for email mass communication.

Despite its age, Email has remained the prevalent form of electronic communication, it's usage being wildly different from how it was imagined. The tools to handle it, however, are still stuck in their original UI metaphors.

Myriad aims at producing personalized communication on a comparable level to manually composed messages, while reducing user effort. A cross-over of mailmerge and customer support/helpdesk software, it enables managing big volumes of bidirectional email-based communication. It is based on a self-developed framework for separating information extraction, decision- making, and personalization steps in communication.

The myriad system consists of a server component that handles interfacing with email servers, a core logic system and a web frontend for users.

It can be tested at <http://myriad.ludwigschubert.de>.

Keywords

Email, Workflow, Helpdesk, Mailmerge, Crowdsourcing, Personalized Communication,
Assisted Templating, Generated Documents, Web-Application

Acknowledgements

This thesis is based on research conducted from October 2012 through April 2013 together with Christian Ikas, Barış Öztop and Nicolas Kokkalis under the guidance of *Prof. Michael S. Bernstein* and *Prof. Scott R. Klemmer* during a stay at Stanford University's Human Computer Interaction Department.

The research stay was partly financed by Elitenetzwerk Bayern through a grant to the study program *Technology Management* at the Center for Digital Technology and Management.



Stanford
University





Contents

Disclaimer	v
Abstract	vii
Acknowledgements	ix
Table of Contents	xi
 Main Matter	 3
1. Introduction	3
1.1. Outline	3
1.2. Motivation	4
1.3. Goals and Background	4
2. Technical Backgrounds	5
2.1. Web Applications	6
2.1.1. Why build a web application instead of a native one?	6
2.1.2. Main Components of a Web Application	6
2.1.3. Rails and Ruby	6
2.2. Email Systems	6
2.2.1. Working with RFC 2822	6
2.2.2. IMAP and SMTP	6
2.3. Workflow Systems	6

2.3.1. History of Workflow Systems	6
2.3.2. Famous Examples	6
3. Comparison with similar systems	7
3.1. Mailmerge Systems	7
3.1.1. CRM Systems	7
3.1.2. Dedicated Mailmerge Systems	7
3.1.3. Backend Services	7
3.2. Customer Support Systems	7
4. Concept	9
4.1. Functional Analysis	10
4.2. Product Functions	10
4.3. User Interface	10
4.3.1. Prototyping Approaches	10
4.4. Technical Analysis	10
4.4.1. Runtime Environment	10
4.4.2. Server Software Stack	10
4.4.3. Client Side	10
4.4.4. Backend Service Connections	10
4.5. System Design	10
4.5.1. Database Schema	10
4.5.2. Distribution of System Components	10
5. Implementation	11
5.1. Preparation and Tools	12
5.1.1. Development Environment	12
5.1.2. Collaborative Development	12
5.1.3. Deployment	12
5.2. Server Component	12
5.2.1. Core Classes and their Interaction	12
5.2.2. Workers and their Jobs	12

5.2.3. Maintenance and Rake Tasks	12
5.3. Backend Services Connection	12
5.3.1. Email Fetching	12
5.3.2. Google Docs Syncing	12
5.4. “Best Practices”	12
5.4.1. Core Extensions	12
5.4.2. Lean Workers with AbstractWorker	12
5.4.3. Monitoring Services	12
6. Evaluation	13
6.1. Comparison with Initial Goals	13
6.2. Observed Use Cases	13
6.2.1. Recruiting Exchange Students for one of the Most Desired Universi- ties of the World	13
6.2.2. Requesting paper Reviews for a Journal	13
6.2.3. Managing Incoming Class Administration Emails	13
7. Conclusion	15
7.1. Conclusion of this work	15
7.2. Discussion of results	15
7.3. Future Work	15
Bibliography	17
List of Figures	19
Appendix	23
A. Selected Source Code Examples	23
B. UML Diagrams over time	25
B.1. Usage of UML Diagrams for internal communication	25
B.2. Stability of UML Diagrams over time	25

C. Colophon	31
--------------------	-----------

Main Matter

1. Introduction

This is the introduction.

1.1. Outline

This chapter will give an overview and arguments for the relevance of new Email tools, as well the goals pursued by this thesis. Additionally, the structure of the work is explained.

In **chapter 2, Technical Backgrounds**, relevant technical background information is provided. The decision to build a web app is motivated. Standard-compliant email systems are explained and relevant standards are introduced. Lastly, an introduction to workflow systems and historic examples is given.

In **chapter 3, Comparison with similar systems**, two major categories of software – dedicated mailmerge systems, and customer support systems – are introduced, which overlap in functionality with Myriad.

In **chapter 4, Concept**, the architecture of the proposed system is motivated and its development process is highlighted.

In **chapter 5, Implementation**, is concerned with technical details of how Myriad was implemented. The collaborative development process is described, as are actual system component and their functionality.

In **chapter 6, Evaluation**, Myriad's initial goals will be reiterated and contrasted with real world observed usage.

In **chapter 7, Conclusion**, the relevance of this system and the proposed framework is discussed; future work is outlined and possible directions proposed.

This is followed by a **Bibliography** of cited works and a **List of Figures**.

The **Appendix** contains selected source code examples, the development of the system design over time in UML Diagrams, and a **Colophon**.

1.2. Motivation

1.3. Goals and Background

2. Technical Backgrounds

2.1. Web Applications

2.1.1. Why build a web application instead of a native one?

2.1.2. Main Components of a Web Application

2.1.3. Rails and Ruby

2.2. Email Systems

2.2.1. Working with RFC 2822

2.2.2. IMAP and SMTP

2.3. Workflow Systems

2.3.1. History of Workflow Systems

2.3.2. Famous Examples

3. Comparison with similar systems

3.1. Mailmerge Systems

3.1.1. CRM Systems

3.1.2. Dedicated Mailmerge Systems

3.1.3. Backend Services

Amazon SES

SendGrid

3.2. Customer Support Systems

4. Concept

4.1. Functional Analysis

4.2. Product Functions

4.3. User Interface

4.3.1. Prototyping Approaches

4.4. Technical Analysis

4.4.1. Runtime Environment

4.4.2. Server Software Stack

4.4.3. Client Side

4.4.4. Backend Service Connections

5. Implementation

5.1. Preparation and Tools

5.1.1. Development Environment

5.1.2. Collaborative Development

`git`, `gitflow` and Github

5.1.3. Deployment

Deployment Tool `Capistrano`

5.2. Server Component

5.2.1. Core Classes and their Interaction

`Contact`

`Template`

12

`Email`

`Campaign`

6. Evaluation

6.1. Comparison with Initial Goals

6.2. Observed Use Cases

6.2.1. Recruiting Exchange Students for one of the Most Desired Universities of the World

6.2.2. Requesting paper Reviews for a Journal

6.2.3. Managing Incoming Class Administration Emails

7. Conclusion

7.1. Conclusion of this work

7.2. Discussion of results

7.3. Future Work

Bibliography

- [1] Leslie Lamport. *LaTeX : A Documentation Preparation System User's Guide and Reference Manual*. Addison-Wesley Professional, 1994.

List of Figures

B.1. The first stable UML Diagram from January 24 th still had separate <code>Email</code> and <code>contacts_messages</code> tables, no rule automation and a semantically incomplete model of Emails which didn't discern between user generated emails and system generated emails.	27
B.2. The second UML Diagram from May 28 th fixed most of the aforementioned problems and already introduced optimizations such as extracting the Raw-Mail content to a different table. Those had become necessary when support for attachments was introduced.	28
B.3. This third UML Diagram from May 29 th finally tamed the <code>Email</code> inheritance tree, and also introduced improvements to legibility for the first time.	29
B.4. By the time of this fourth UML Diagram from April 18 th only minor improvements were still made to the system design, such as adding <code>ValueSettingActions</code> and supporting assistants via <code>SharingAssignments</code>	30

Appendix

A. Selected Source Code Examples

```
1 class Notification < ActiveRecord::Base
2   belongs_to :user
3   attr_accessible :user, :message, :resource_id, :resource_type
4
5   def resource
6     @resource ||= resource_class.find resource_id
7   rescue ActiveRecord::RecordNotFound
8     nil
9   end
10
11  def resource_class
12    resource_type.classify.constantize
13  end
14
15  def has_resource?
16    should_have_resource? and resource.present?
17  end
18
19  def should_have_resource?
20    resource_type.present? and resource_id.present?
21  end
22
23 end
```


B. UML Diagrams over time

Test esetnetcyiegsllcsgeycagfjsgf

B.1. Usage of UML Diagrams for internal communication

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

B.2. Stability of UML Diagrams over time

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend

at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

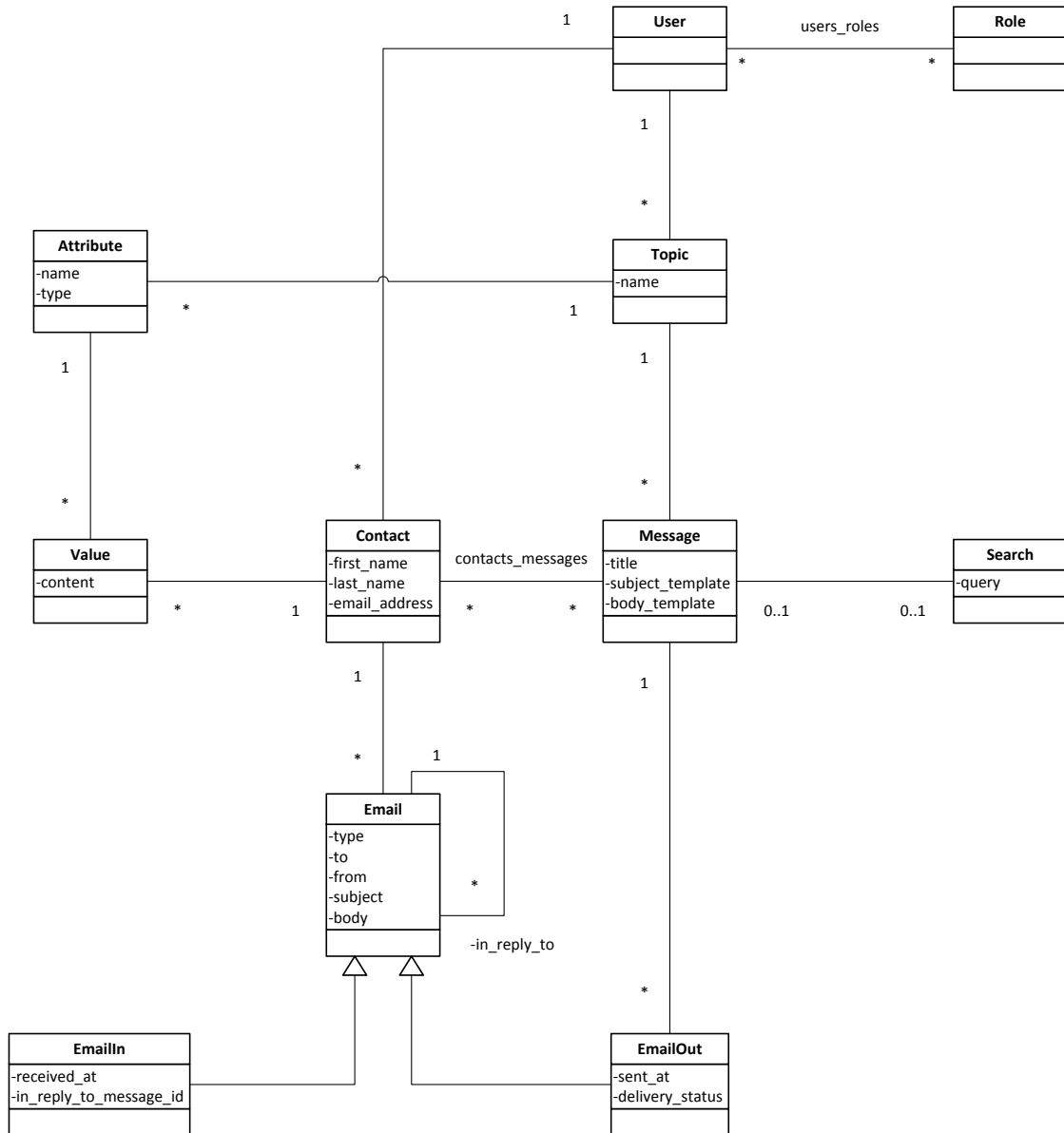


Figure B.1.: The first stable UML Diagram from January 24th still had separate `Email` and `contacts_messages` tables, no rule automation and a semantically incomplete model of Emails which didn't discern between user generated emails and system generated emails.

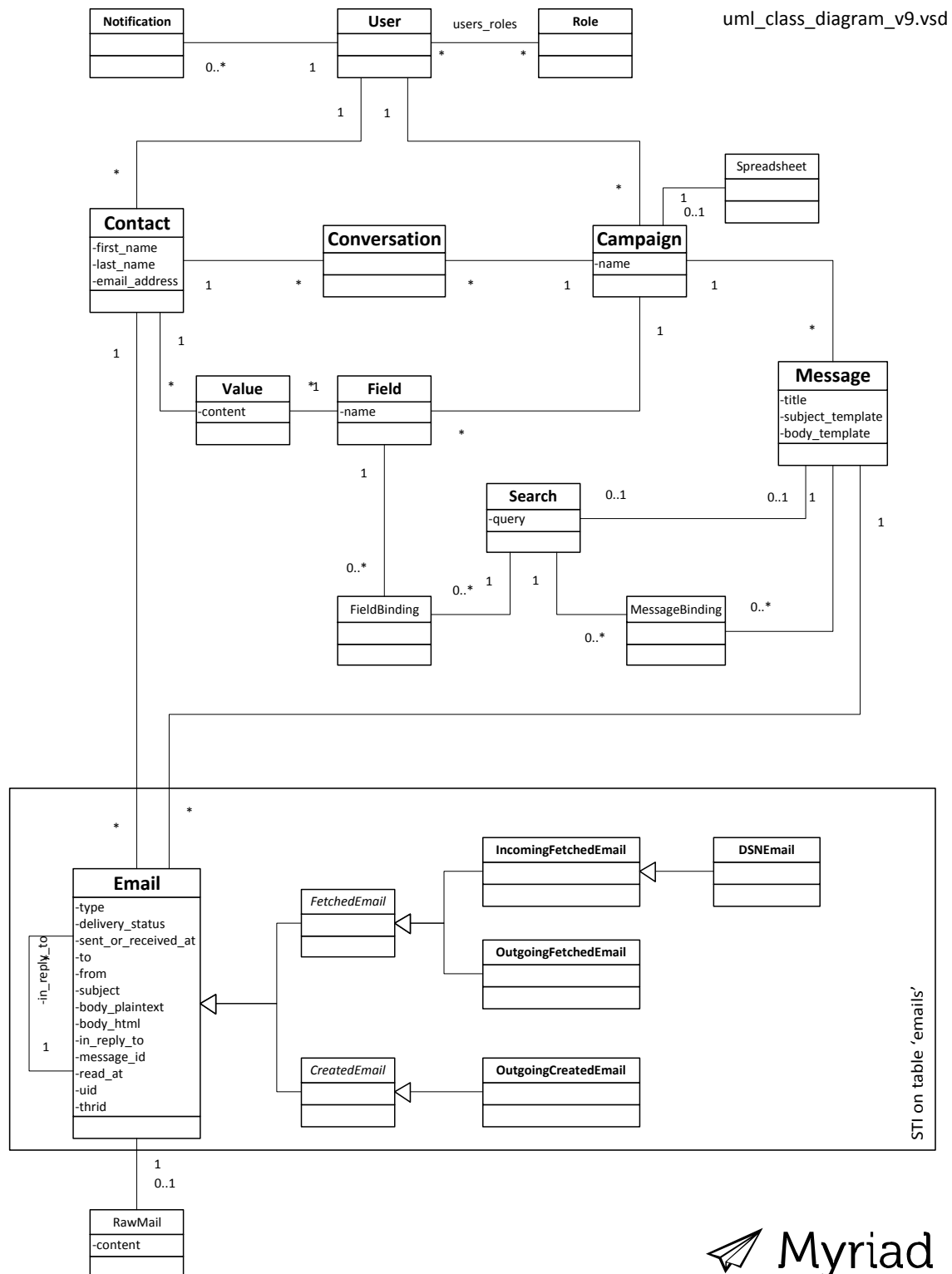


Figure B.3.: This third UML Diagram from May 29th finally tamed the `Email` inheritance tree, and also introduced improvements to legibility for the first time.

B. UML Diagrams over time

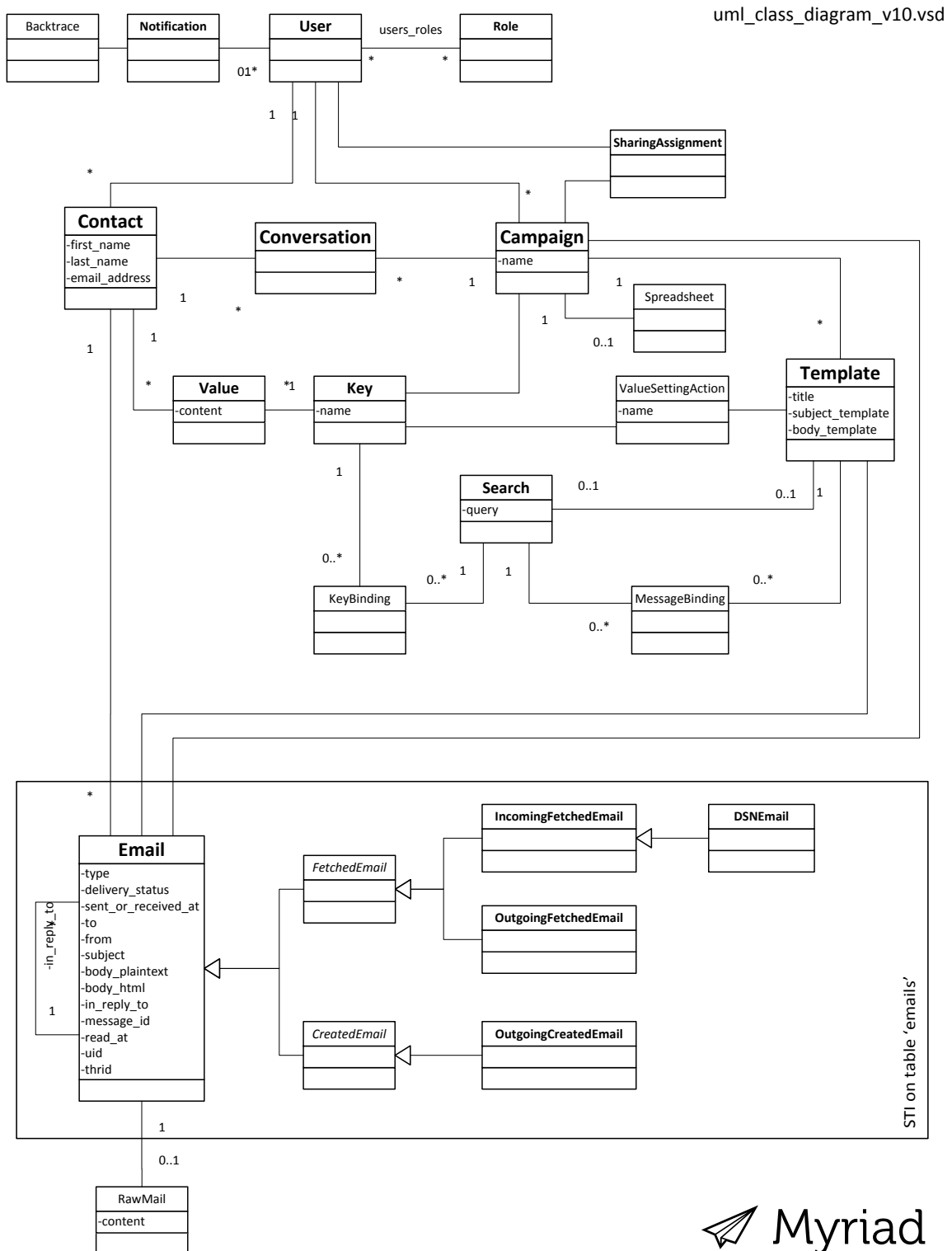


Figure B.4.: By the time of this fourth UML Diagram from April 18th only minor improvements were still made to the system design, such as adding ValueSettingActions and supporting assistants via SharingAssignments.

C. Colophon

This thesis is set in \LaTeX [1]. The template used is based on the official TUM Computer Science template. The sources are hosted publicly on GitHub, while the actual PDF file is built by the continuous integration server Travis.