Mail::Box – Final Report

Mark A.C.J. Overmeer*
MARKOV Solutions, Arnhem

7th February 2003

1 Introduction

`Mail::Box` is a module for the Perl programming language. It helps programmers to process e-mail message automatically. For instance, `Mail::Box` can be used to filter incoming unsolicited messages (*spam* and viruses), create automatic replies to costumers, search through mail archives, or produce web-based mail applications. `Mail::Box` is not a user application itself, only a library which facilitates application writers.

`Mail::Box` was created by me, Mark Overmeer, in 2001 as a modern replacement for existing e-mail handling libraries for Perl. It had already grown to quite a large module, in need for a push to market.
The NLnet Foundation offered to support the development of `Mail::Box` for a few months, in which a set of improvements had to be made. These improvements targeted both the implementation and the building of a user's community. This short paper describes the improvements which were realized.

2 Promotion

The main focus for this NLnet funded project was to enlarge and organise the user community. A module which has more users will get better, and therefore attract even more users, where a module which does not actively promote itself will eventually diminish. Besides, the module required more involved developers to safeguard continued improvements.

A number of promotional activities were launched:

- One presentation and paper at the SANE2002 Conference in Maastricht.

*e-mail: `mark@overmeer.net`, web-site: `http://mark.overmeer.net`

- One tutorial and a paper on YAPC::Europe, the yearly European Perl Conference, this time in Munich Germany.
- A web page which is used as central development location, listing all `Mail::Box` resources at `http://perl.overmeer.net/mailbox/`
- A mailing-list `mailbox@overmeer.net`, with (on feb 1st, 2003) 70 members. The mailing list started at May 20, 2002. From then on, in eight months, it processed about 1500 e-mail messages. This is 4 times the number of messages I sent and received related to the module in the eight months before the list started.
- Most of the mailing list messages are saved in the archive at `http://marc.theaimsgroup.com/?l=perl-mailbox`. Some discussion threads were not broadcasted to everyone, to avoid list members from being over-flooded with less interesting debates. These messages can not be found in that archive.
  Next to the standard mailing list, some regular contributors have their own direct link to me, as main implementor. Messages on some `Mail::Box` related modules are kept separate as well. A simple message count reveals about 300 messages more.
  This total of 1800 messages in eight months consumed a considerable amount of the project time.
- Mail::Box had two items on the central Perl story board website, located at `http://use.perl.org`.

Seen the enormous (and growing) amount of messages, it is clear that the number of users is rising.

3 Quality improvements

One of the main targets for the project, was to improve the quality of the library. Based on the algorithms of the existing (old) modules, `Mail::Box` had to be improved to fulfil the requirements of the latest RFCs.

3.1 Conformance

Code improvements have been made in many areas, for instance in the strict protection of the order of header lines, the improved processing of *resent-groups*, and the handling of messages which are not RFC compliant. These improvements were mainly possible due to the growing amount of users, giving feedback on what they found.

3.2 Documentation

The larger a library gets, the harder it is for users to understand what is happening inside them. `Mail::Box` defines over 800 methods in 110 classes, which is not easy to grasp.

During this project, the organisation of the documentation is re-done. The manual-pages are now partially generated, to help the user understand class dependencies. Available methods are grouped, sorted, and now have a consistent layout. To achieve this, the standard POD document feature of Perl was extended from a visual style mark-up into a logical style mark-up language. It would be nice to publish this extension to benefit the whole Perl community, however the time to publish this feature is lacking.

Not only the structure of the manual-pages is improved: they are now also available in HTML format, which certainly helps to browse through them. The HTML pages contain exactly the same information as the pages which are available through the `man` command. Generating documentation especially for browsing would improve the accessibility of the information even further.

3.3 Tests

Distributed with `Mail::Box` are 75 test scripts which define over 7000 different regression tests. During the time-span of this project, these tests have been re-organised into groups to facilitate tests on optional extensions to the library. Failure in some groups of tests will break the installation of the module, where other tests are considered less important.

The output of the tests is improved as well: failing tests will report what went wrong, in stead of just fail. This simplifies solving the rare installation problems which some users experience.

3.4 Portability

A lot of effort has been made to resolve all problems with the tests when run on Windows. The library is platform independent, but some of the tests are not. Especially differences in representation of line-endings (LF character for UNIX, CR/LF characters for Windows, CR for the old Mac) caused many tests to break.

Nearly all these problems were solved. There are some differences between versions of Windows, for example the level of POSIX compliancy, which still cause failures; this still has space for improvements. Most portability issues to Windows were solved with a sustained effort by Greg Matheson (an American guy working in Taiwan).

4 Extensions

Many extensions to the module have been realized. Most of them were planned, but some where initiated on user's request.

I have tried to get people involved by including their code. This was a much harder task than expected: even experienced programmers need time to work through a library of this size. In a salaried job, you usually can allocate some hours to learn, but when you have to do developments in your own time, this is usually too much of a burden. During the whole period of the project, people showed good intention to contribute, however this rarely succeeded.

The most important feature which was planned to be implemented by someone else was IMAP. IMAP is a very complex asynchronous protocol, which really demands a lot from the implementor. Too much, as came clear during the project. A few alternative IMAP implementations were discussed, and attempts were made to challenge someone to code it. Preparations were made to simplify embedding IMAP into `Mail::Box`. But all efforts were in vain: implementation of IMAP is a serious task, which is too much work for voluntairy contributors.

Realized major extensions:

- Liz Mattijsen wrote a POP3 extension. This extension was made in a way that it does not matter for a application programmer whether the messages are located on a local disk with a file based message format, or available via POP3. Even breaking connections will not bother the user's application.
- Greg Matheson wrote a connection to Exim, an open source mailer.
- Together with Edward Wildgoose, strict reporting of status codes in raw SMTP protocol handling was implemented.
- A connection was made to Spam Assassin, a well known spam filter.
- As planned, a message parser written in C was published. One reports speaks of a four-fold speed improvement when opening message folders.

The number of extensions, fixes and improvements is too large to list here. The change log reports 261 changes made during this project. These changes where contributed or initiated by 42 different people. In most cases, I had to write the patch.

5 Releases

The project has maintained a “release often” policy, to pass improvements as fast as possible to all users. `Mail::Box` is the name of the main module which is covered by this project, but there are a few other modules involved. These modules are distributed as separate packages because they have a use outside message processing.

The full set of related modules, which are maintained by me:

`Mail::Box`
: By far the largest module, defining 90 classes to handle mail messages and mail folders. It has seen 24 releases during this NLnet funded project.

`Mail::Box::Parser::C`
: The message parser which is written in C, is distributed as separate package: some people (especially on Windows) do not have a C compiler available. The main package does not require a C compiler at all, where this extension needs one.

`MIME::Types`
: Defines MIME types: standardised descriptions of data types. It has seen 6 releases.

`Object::Realize::Later`
: A tricky module which helps `Mail::Box` to perform with a reasonable speed. It got special attention during the YAPC::Europe conference, and is slowly getting broader acceptance.

`User::Identity`
: A new module, which tries to standardise the handling of general user information, like name and address. This data can be retrieved from various sources (LDAP, database), and used on many places (for instance to construct e-mail messages). This module has just seen its first alpha release.

`Mail::Identity`
: In combination with a `User::Identity`, it stores the various roles a person can have when processing e-mail. For instance, you may have role as company representative, private person, and web master. It combines information like reply address, pgp-key, and signature for each role. Not released yet.

`MailTools`
: This set of modules are to be replaced, but are still maintained. This

bundle of modules is marked deprecated: only bugs are fixed. MailTools was hit by a security alarm, discovered by SuSE. It has seen 12 releases during this NLnet project. The number of requests for supports is decreasing.

6 Conclusions

The `Mail::Box` module is more and more used, and the older e-mail related modules have less and less users. The only disadvantage of this growth is the increasing call for support. It consumes at least one hour a day to help out and fix bugs or implemented small requested features.

The IMAP implementation did not succeed, which is a pity. On the other hand, a POP3 implementation has been realized unexpectedly. A speed up when using a C-based message parser is also achieved.

`Mail::Box` has more users and a user community, better documentation, more conforming implementation, and more features. The project was very successful. About ten statements of support by users are available as well.

The wish-list of features waiting to be implemented is long. Plans for continued development as discussed in a separate paper.