

Task-Oriented Restructuring of an Application Domain: A Multi-Agent Architecture for Doing Things in Internet E-Mail

Juha Takkinen and Nahid Shahmehri

Laboratory for Intelligent Information Systems (IISLAB),

Department of Computer and Information Science, Linköping University, Sweden

juhta@ida.liu.se and nsh@ida.liu.se

Abstract

Users are increasingly more mobile, using different e-mail systems in different places. Based on the results of an explorative study of task management in a user's daily electronic life, and earlier work on information management in e-mail, we begin a restructuring of the domain of Internet e-mail. The purpose is to make it even more suitable for message-based communication in organizations, where many users, groups, and tasks are involved. By introducing the concept of tasks we hypothesize that e-mail can be made more usable for managing short-term interests of a user. We describe a design of a multi-agent architecture that supports the integration of disparate and distributed information sources. Our goal is to combine the information in the sources for the dynamic and automatic construction of classification and prioritization rules. A prototype implementing the architecture is tentatively described, which uses information from the user's current electronic environment to manage her short-term interests.

1. Introduction

E-mail is extensively used for cooperative communication and work in both virtual and real organizations. The most commonly available software applications for managing e-mail today can provide a user with much value. For example, the e-mail client (the e-mail reader software) provides basic functionality to manage messages [24]: filtering (classifying and prioritizing) messages based on keywords found in message headers is common, as is the use of folders to store messages, and two-paned and three-paned display of folders and messages on the screen.

However, users with high mail traffic (say, 50–100 messages per day) and who also travel a lot and/or are associated with several different virtual or real groups experience problems. Typical problems include different functionality in the e-mail client and different rule sets for classifying messages, which both often depend on the current workplace. The electronic environments are

different and so is the user's role in each place or organization. The mostly unstructured messages with different kinds of information content, interaction patterns that can be very complex, and interaction partners that come and go that characterize e-mail, all imply respectively a strong need for support for more efficient management of e-mail messages.

While keyword-based filtering can be used to model a user's *long-term filtering needs*, it is not satisfactory for modelling *the short-term interests* of a user, such as getting updates to meeting agendas or hearing back from people one has written to recently. Furthermore, in rule-based, user-driven filtering, where each user is responsible for filtering her own messages, the writing and updating of the filtering rules can be both complicated and time-consuming [16]. A user's short-term interests are typically represented by rather concrete tasks that the user is performing, such as negotiation (deciding upon a meeting's place, time, and contents) and information request. By introducing the concept of tasks we hypothesize that e-mail can be made more usable for managing short-term interests of a user.

The information about a user's concrete daily tasks is available implicitly in different formats in the electronic environment. The information is stored in applications and sources ranging from the outgoing e-mail messages to the calendar manager and the address book/rolodex manager. The information can even be located on another computer in the network. This distributed nature of the information calls for a special kind of architecture. It should be possible to automatically coordinate applications and sources, and still keep the simplicity of e-mail use (composing and sending messages, reading, deleting, and storing messages, etc.). The distributed solution that we propose to use for modelling and implementation is *agents*. We also make *learning* a central component in the classification process.

The remainder of this paper is structured as follows. We begin with a background. We then present our agent architecture, which is based on a previously developed framework for task management in e-mail [25] and a continuation of our previously developed conceptual model

for information management in e-mail [24]. We conclude the paper with conclusions and future work.

2. Background

2.1 An explorative study

Most users have developed their own strategies for handling e-mail [21]. Many also know how to define simple filtering rules based on keywords to classify messages, although it can be time-consuming and awkward [22]. This keyword-distribution-based approach is almost as effective as more sophisticated approaches [2][20]. However, the rules do not change to reflect daily or weekly activity, i.e., short-term interests [16].

In an explorative study we found that users are becoming increasingly mobile [23], which indicates that different rules are needed for different electronic environments and roles. A task can be defined as an exchange of more or less naturally chained messages. Small tasks can be contained within a thread of messages (usually with the same subject). A thread is a series of messages that are linked via the **In-Reply-To** or **References** headers to a specific root message (the initial message) [3]. Note, however, that the same **Subject** can occur in two different threads out of pure coincidence. Also, certain types of messages dominate certain tasks. For example, the action memo, or a Policy message [4] is typically used for delegating tasks within a project [6]. Hence, threads and message types can be employed to keep track of tasks. Furthermore, since the user may employ synchronous communication (e.g., telephone or ICQ [10]) in combination with e-mail, there has to be some way of explicitly marking a task (a thread) in e-mail as completed. This function mimics the kill files used in netnews.

2.2 An open system

Internet e-mail users have a common problem of confirming the deliverance of a message. This is a problem associated with the open architecture of Internet e-mail. Here, standards can be used—IETF is working with the introduction of message disposition notifications (MDNs) into the message header [5], which partly can solve the problem with delivery confirmation.

One of our main requirements is to keep the simplicity of use and open architecture of Internet-based e-mail from a user's and a developer's viewpoint, respectively. An open architecture or system is characterized by the dynamic changes of system structure [13]: components are not known in advance, they change and they are heterogeneous (implemented by different people, at different times, using different software tools and techniques). Legacy components need to interact: many different kinds of e-

mail clients are available [24], a single user can be employing a different e-mail client depending on the place [25], and different sources are available to a user and her e-mail client depending on the computer account being used. In an open system with these characteristics, agents alleviate the work of both a developer and a user [13].

2.3 CAFE—our starting point

Previously, we have developed CAFE (a Categorization Assistant For E-mail), a conceptual model for information management in e-mail [24]. The user can explicitly tell the system her state of mind via three modes (fig. 1). The Cool

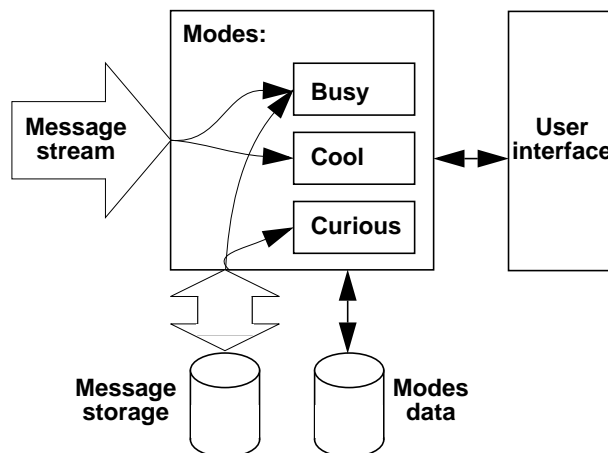


Fig. 1: The conceptual model of CAFE.

mode (default) contains the traditional filtering of messages, including the user's own folders. The Curious mode is invoked when the user has much time on her hand and she wishes to organize messages from, for example, a mailing list. The Busy mode is invoked at times of high stress and makes use of system-defined folders to organize new and unseen messages in priority order. An implementation of CAFE [24] required using a different information retrieval and filtering technique for each mode.

We are now extending CAFE, concentrating on the Busy mode, with the ultimate goal to refine and extend each of the modes in CAFE. In the Busy mode we have all new and unseen messages—and the interesting problem of modelling a user's short-term interests. We are specifically interested in messages that a user puts on her to-do list, since these are often connected to tasks. While the Busy mode in CAFE classifies messages into prioritized folders, we now refine and extend it to classify messages according to a user's current tasks. In other words, messages that belong to the same thread or conversation are grouped and made available to a user according to her current, short-term interests. Furthermore, to model the short-term interests, information in other relevant applications and

sources are also taken into account, and not only information in the form of e-mail messages.

3. An agent architecture

Our approach with accessing different electronically available sources (see previous section) in a user's environment presents at least three complications [8]:

- *distribution*: not every classification can depend on data from one single source; fragments of useful information need to be combined
- *heterogeneity*: different sources may be using different access languages and protocols; for example, different words may be referring to the same concept
- *instability*: existing sources may be changing the format of their data, or changing their content.

So-called intelligent agents [13][21] are typically constructed to alleviate these complications. An example of a system that uses agents is Infomaster [8], which is a generic information integration tool for integrating existing information sources. For each type of information source (legacy component) a special program called *agent wrapper* is constructed that translates between the "language" of the information source and the core system [13]. The internal content language used by Infomaster is the *knowledge interchange format* (KIF) [7][15], a lisp-like language used for representing first order logic expressions. An example of a system utilizing "intelligent agents" before the concept was widely known is Information Lens [17], which helped users intelligently find, filter, sort, and prioritize e-mail messages. A hierarchy of semi-formal templates are used to structure messages. A multi-agent architecture transfers the burden or interoperation from the developers and users of programs to the programs themselves. This gives the illusion of centralization and homogeneity of information and makes the system flexible and scalable [7].

Our system for classifying e-mail messages (see fig. 3) is inspired by Infomaster's multi-agent architecture. Two common and heterogeneous information sources are shown in the figure: the calendar manager and the outgoing e-mail messages. The facilitator is a special system program that coordinates the activities of the agents on one machine in a network. The interface of the e-mail client has an *interface wrapper* or interface agent that translates messages from and to the application/source agents. A *rule agent* is also shown in the fig. 3. The *learning agent* combines the information from the sources in a user's electronic environment (the outgoing messages and the calendar in fig. 3), and any refilings of messages that a user does, to create and update classification rules for incoming messages as well as different (logical) views in the interface. The views are managed by the interface agent.

The classification process is shown in more detail in fig. 4. A new source (consisting of legacy code) is added to the system by writing a wrapper (see fig 2) and introducing it to the *facilitator* (the wrappers are the shaded ovals in fig. 3). The Shade KAPI in the figure is the interface to the agent messaging language [11]. In other words, a wrapper

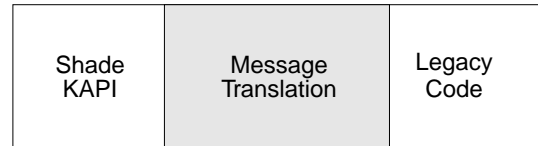


Fig. 2: Wrapping legacy software.

injects code into a program and makes it communicate in the agent messaging language.

There are three more basic agents in our architecture. To save space they are not shown in fig. 3 but instead shown in fig. 4, as part of the classification process: the information filtering (IF) agent, the information retrieval (IR) agent, and the personal database (PDB) agent. The PDB represents a user's collection of previously received and stored messages, not necessarily stored on one single computer.

4. Towards a prototype: a worked example

In a prototype new and unseen messages are first parsed for "interesting" words, such as for example proper nouns and words in header and body that are not stopwords. This parsing is performed by the information filtering (IF) agent (see fig. 4). In this way message surrogates are created, together with an index. The other agents managing the outgoing messages and the data in the calendar (and any other sources) respectively also create indices. Because of computational reasons [18] the latter indices are created only once per day.

The learning agent combines the information in the indices (not shown in fig. 4) over new and unseen messages, and over the electronic sources, to automatically and dynamically construct the classification rules. Two learning processes are distinguished:

- learning rules for classifying messages into folders
- learning rules for prioritizing messages in folders.

Note that so-called *negative examples* are taken into account in both learning processes, i.e., when a user refiles one or more messages that have been improperly classified by the learning algorithm, the rules are updated accordingly. This is done so that future messages can be more correctly classified. The action of the user manually refiling messages is represented by the arrow from View to Personal Database in fig. 4.

A user's task is initialized at the user's request in either of two ways:

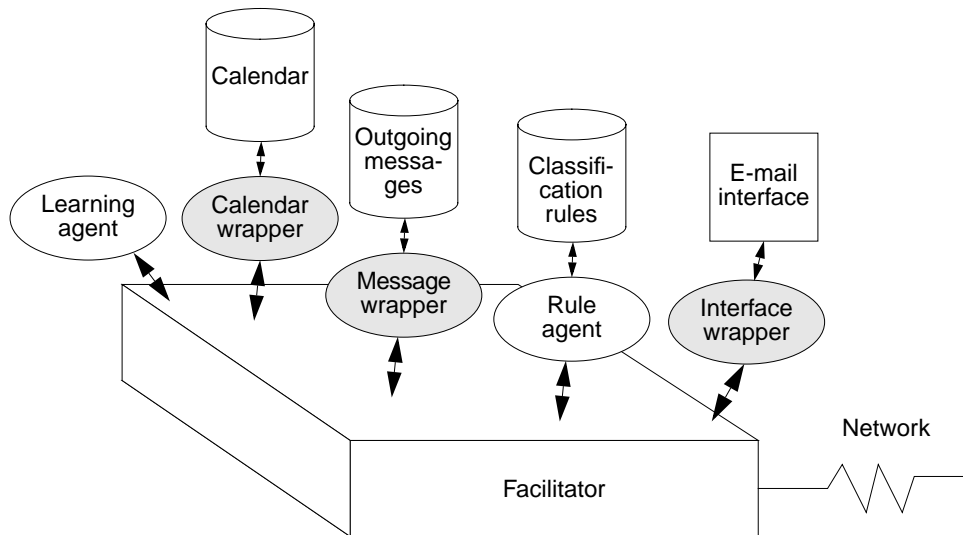


Fig 3: A multi-agent architecture for information integration in e-mail.

- The user replies to an existing and already indexed message. The message and the reply to it are both placed in a separate folder.
- The user composes a new message using a simple template [16] representing the wanted task. The message is moved to a separate folder representing current tasks of the user.

In these ways the user explicitly tells the system (via the interface agent) to keep track of the commencing exchange of messages that is going to represent the task. In both cases the messages (the reply and the new message respectively) are also logged, i.e., copied into the folder for outgoing messages. Furthermore, the classification rules are updated with information about the messages and their storage places.

When formatting a message for presentation to the user,

features (header fields) that are more important for the current type of the message are reinforced (or suppressed, if the features are less important) by the interface agent. Furthermore, messages in threads representing tasks are indented, starting from the root message that initiated the task.

The user also explicitly tells the system when a task is finished by marking a message in the task and by, for example, pressing a key. The classification rules are updated and the learning agent “unlearns” what it knows about the task that has been finished.

At the user’s request, the interface agent would also possibly give feedback to the user about how and why a message has been classified and prioritized in relation to a task.

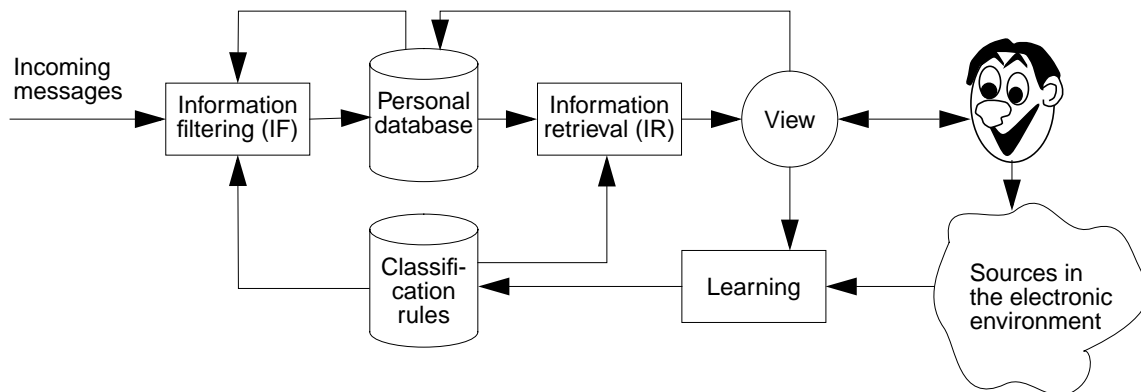


Fig. 4: The classification process.

5. Conclusion

Applications that can cooperate, coordinate and share their information with other applications, with or without user intervention, are in increasing demand. This is especially true in the distributed electronic world that we live in today.

A user's short-term interests can dynamically be modelled (and without extra effort from the user) by using the data available in the user's electronic environment (cf. [18]). Some examples of a user's electronically available data sources are:

- messages in the user's to-do folder (if it exists)
- the outgoing messages
- the address book
- the calendar tool
- the bookmarks of the web browser
- the database over employees in the organization.

We use an agent-based approach since agents enable the autonomous, dynamically adaptive and robust operation and interoperability among software systems [9], which is needed for our management of the different applications available in the electronic environment of a user. Furthermore, there is a set of semantically standardized interactions between static software systems provided in certain agent communication languages.

To summarize, the agent approach satisfies our needs specifically because:

- Internet e-mail is an open system
- we need techniques for negotiation and cooperation
- data and resources are distributed.

We have also made the idea of learning central to the classification process. Learning supports the high degree of adaptivity required in the special domain of e-mail while minimizing user intervention [20]. We have strived to follow three general requirements for the classification part of our system architecture—it should be:

- *automatic*: a user should not need to create classification rules manually
- *dynamic*: classification rules are updated accordingly when the electronic environment changes
- *adaptive*: the system improves with experience.

6. Future Work

We are now implementing a first, simple prototype of the agent architecture in Java [12]. We are considering the use of JATLite [11] (Java Agent Template, Lite) as a basic infrastructure for our agents. JATLite is a package of programs written in the Java language that allow users to

quickly create new software "agents" that communicate robustly. We are also examining machine learning algorithms [2][19] and how to combine them with domain knowledge in the system.

To access the meanings and contents of messages we intend to make use of speech act theory (SAT) [1]. An early e-mail-based system that helped people structure conversations and track tasks using SAT is Coordinator, described by Winograd and Flores [26]. We intend to make use of SAT as a "container" for the task-specific information in a message, and at the same time keeping the simplicity of e-mail. The interface agent can accomplish this by using basic templates for detecting message types. The relationship between messages, templates, types, and tasks is shown in fig. 5. Message templates form the basis

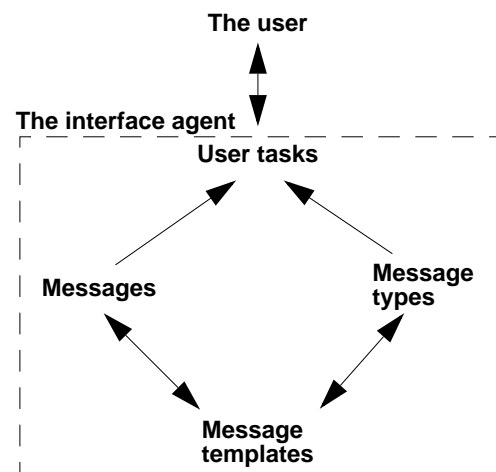


Fig. 5: The relationship between messages, templates, types, and tasks.

for giving structure to messages and for characterizing message types. The messages and the message types can in turn be used to characterize tasks and to find matching message templates.

We are looking at defining a version of FLBC, a Formal Language for Business Communication described by Kimbrough and Moore [14]. As argued in [14], it would be wise to attend to SAT if we are to develop general message handling systems.

Acknowledgments

This work is supported by the Swedish Foundation for Strategic Research. We thank the anonymous referees of our paper for their constructive comments.

References

- [1] Austin, J. L. (1962), *How to Do Things with Words*. London, U.K.

- [2] Cohen, W. W. (1996), "Learning Rules that Classify E-Mail," in *AAAI Spring Symposium on Machine Learning in Information Access*, Stanford, March 25–27, 1996.
- [3] Crocker, D. (1982), "Standard for the format of ARPA Internet text messages," *RFC822*, August 13, 1982. Updated by RFC1123, RFC1138, RFC1148, RFC1327.
- [4] Crowston, K., Malone, T. W., & Lin, F. (1988), "Cognitive Science and Organizational Design: A Case Study of Computer Conferencing," in *Human-Computer Interaction, 1987-1988*, Vol. 3, pp. 59-85.
- [5] Fajman, R. (1998), "An Extensible Message Format for Message Disposition Notifications," Internet draft, February 6, 1998. <ftp://ftp.nordu.net/internet-drafts/draft-ietf-receipt-mdn-08.txt>
- [6] Fleming, S. T. & Kilgour, A. C. (1994), "Electronic Mail: Case Study in Task-Oriented Restructuring of Application Domain," in *IEE Proceedings: Computers and Digital. Techniques*, 141(2), March 1994, 65–71.
- [7] Genesereth, M. R. (1997), "An Agent-Based Framework for Interoperability," in J. M. Bradshaw (ed.) *Software Agents*, 317–345. AAAI Press. ISBN 0-262-52234-9.
- [8] Genesereth, M. R., Keller, A. M. & Duschka, O. (1997) "Infomaster: An Information Integration System," in *Proc. of 1997 ACM SIGMOD Conference*, May 1997.
- [9] Gustavsson, R. (1998), *Personal Communication*. The Societies of Computation (SoC) research project. The University College of Karlskrona/Ronneby, Sweden.
- [10] - (1998), *ICQ—World's Largest Internet Online Communication Network*. <http://www.mirabilis.com/>
- [11] - (1998), *JATLite*. <http://java.stanford.edu/>
- [12] - (1998), *Java Home Page*. <http://java.sun.com/>
- [13] Jennings, N. R. & Wooldridge, M. J. (Eds.) (1998), *Agent Technology*. Springer Verlag.
- [14] Kimbrough, S. O. & Moore, S. A. (1997), "On Automated Message Processing in Electronic Commerce and Work Support Systems: Speech Act Theory and Expressive Felicity," in *ACM Transactions on Information Systems*, 15(4), Oct. 1997, 321–367.
- [15] - (1998), *Knowledge Interchange Format (KIF)*. The Logic Group, Stanford University. <http://logic.stanford.edu/kif/kif.html>
- [16] Malone, T. W., Grant, K. R., Lai, K-Y, Rao, R., & Rosenblitt, D. (1987), "Semistructured Messages Are Surprisingly Useful for Computer-Supported Coordination," in *ACM Transactions on Office Information Systems*, 5(2), April 1987, 115-131.
- [17] Malone, T. W., Lai, K.-Y., & Grant, K. R. (1997), "Agents for Information Sharing and Coordination: A History and Some Reflections," in J. M. Bradshaw (ed.) *Software Agents*. 109–143. AAAI Press. ISBN 0-262-52234-9.
- [18] Marx, M. & Schmandt, C. (1996), "CLUES: Dynamic Personalized Message Filtering," in *Proceedings of CSCW'96*, 113-121.
- [19] Mitchell, T. M. (1997), *Machine Learning*. McGraw-Hill. ISBN 0-07-042807-7.
- [20] Mostafa, J., Mukhopadhyay, S., Lam, W., & Palakal, M. (1997), "A Multilevel Approach to Intelligent Information Filtering: Model, System, and Evaluation," in *ACM Transactions on Information Systems*, 15(4), Oct. 1997, 368–399.
- [21] Shoham, Y. (1993), "Agent-Oriented Programming," in *Artificial Intelligence*, 60(1), 51–92.
- [22] Takkinen, J. (1994), *CASUAR – en prototyp av ett användargränssnitt med filtrering och automatik för hantering av elektronisk post*. In Swedish. M.Sc. Thesis, Linköpings universitet, Sweden.
- [23] Takkinen, J. (1998), *Fallstudie av uppgiftslösning i e-post: resultat*. In Swedish. Internal memo. IISLAB, Dept. of Computer and Information Science, Linköpings universitet.
- [24] Takkinen, J. & Shahmehri, N. (1998a) "CAFE: A Conceptual Model for Managing Information in Electronic Mail," in *Proc. of 31st Hawaii International Conference on System Sciences (HICSS-31)*, Jan. 6–9, 1998, Honolulu, Hawaii, USA.
- [25] Takkinen, J. & Shahmehri, N. (1998b), "Delegation of Tasks and Dissemination of Information in Organizations: Restructuring Internet E-Mail for Doing Things," in *Proc. of American Information Society 1998 Americas Conference*, the Group Supported Collaboration mini-track, Baltimore, Maryland, Aug. 14–16, 1998.
- [26] Winograd, T. & Flores, F. (1986), *Understanding Computers and Cognition: A New Foundation for Design*. Norwood, N. J.: Ablex.