Ad-hoc Workflows for Higher Education

David Martinho[1], Samuel Coelho[1], Luis Guerra e Silva[1], João Carvalho[1], Rita Severo[1], Luis Cruz[1], Artur Ventura[1], Pedro Santos[1], Ricardo Barata[1]

[1]Técnico Lisboa, Avenida Rovisco Pais, 1049-001 Lisboa, Portugal
hello@fenixedu.org

Keywords
Workflow, Ad-hoc, Timeline, Information Systems, Innovation

ABSTRACT

It is commonplace in higher education institutions, for a large number of informal business processes to be handled through e-mail. However, e-mail fails to properly support traceability, and it fosters artifact duplication. This paper introduces GEARS: a solution to the e-mail overuse phenomenon. GEARS is a new approach to ad-hoc workflow systems that focuses on keeping the simplicity of the e-mail user experience while implementing a participation-driven process execution.

1. INTRODUCTION

Nowadays, higher education institutions usually resort to workflow systems to automate business processes that are well-structured and have a large user base. However, a significant number of other processes is usually executed by e-mail [1], either because their structure is not clearly defined and amenable for implementation in a typical workflow system, or because the small number of users does not render such implementation cost-effective. The bulk of these informal business processes starts whenever someone needs something, in which case he/she sends an e-mail to another person who may or not delegate part of the request and handle the other part. Although many of these e-mail interactions regard authoritative and informative actions, other e-mails result in the production of data artifacts.

Most often, business processes are handled informally via e-mail instead of workflow systems due to their dynamic and loosely structured nature. There have been research works that focus on uncovering interaction patterns and processes by mining e-mail logs [2]. Dynamic processes are too complex to be implemented in a system that enforces a given behavior because the behavior has many exception cases. Furthermore, the effort of implementing such rules is overkill when the process rules change and the existing model is no longer adequate to the process requirements.

One of the main reasons that e-mail is used to support such dynamic informal processes is that e-mail is a general-purpose communication tool, generic enough to adapt to the most subtle business process changes: there is no underlying rule model except request and a possible response. However, the traceability that e-mail provides is inadequate for business process execution. Also, the amount of replication and dispersion of data artifacts among the participants hinders an efficient execution of the process. Finally, the execution of a given business process can be scattered among different e-mail threads with different e-mail subjects. There is no central system where all the interactions, subjacent to the execution of a given process, are handled.

This paper is organized as follows. In Section 2, we present a case study that serves as an introduction and motivation for the contributions proposed in this paper. Section 3 introduces the proposed system as a novel approach to tackle the problem of e-mail overuse in loosely structured business processes. Afterwards, Section 4 describes relevant implementation details. Finally, Section 5 presents a few concluding remarks and highlights topics for future work.

2. A CASE STUDY

In Técnico Lisboa, a higher education institution in Portugal, the most relevant academic and administrative business processes are already automated using the FenixEdu[1] Education Management System. However, either due to their dynamic nature or because they are of interest only to a small portion of the school's population, a wealth of other business processes is still informally executed resorting to e-mail. We confirmed such behavior when we conducted an informal survey on a small group of faculty and administrative staff.

One of the first processes that we identified, within the informal survey, as being highly dynamic, and mostly based on e-mail, are all the interactions and decisions of the various committees within each department. Departmental committees are groups of faculty that either coordinate or provide guidance on different aspects of the daily operation, governance and strategy of each department. Either because each committee addresses issues of very specific nature, or because each department has a distinct culture, departmental committees tend to operate in a very diverse and heterogeneous manner. Therefore, departmental committees tend to have different workflows through different sets of interactions among different members. Usually, such processes are executed via e-mail, where the participants share their opinions through comments, while simultaneously providing and building documents. Nevertheless, the complexity associated to these processes is too high to consider them as case study of the approach that we are proposing in this paper.

We focus our case study in the Master Thesis process, which is also a good example of a process that we found hard to implement in a rigid workflow due to its dynamic nature. To begin with, Master Theses have a varying number of committee members. Also, apart from the advisor, the candidate may have a co-advisor, which would trigger a set of different interactions between them. During the process, the candidate may change the dissertation topic, maintaining or not the advisors. Depending of the situation, the process needs to consider different steps. The same happens during the defense phase: the student may fail to defend the thesis, or the jury may change according to rules like: the candidate must not share publications with any member of the jury besides his supervisors. There is high variability during the different phases of the process. Also, every year, the rules tend to change to adapt to unpredicted situations that were faced in the past year(s) or simply due to external governance events. An interesting aspect is also the fact that, in parallel with the execution of the process, several of the actors involved in the process will most likely need to exchange information (notes, files, comments, etc), and typical structured workflow systems do not provide a practical method to store such information in a format that is easy to later retrieve and access.

3. THE NEW APPROACH

Given the dynamic aspect normally associated to informal processes, and the clear disadvantages in using e-mail to support them, we implemented GEARS, which introduces a new approach to their execution. Commonly, workflow systems either drive process execution through activities or data. However, both approaches require pre-specification of the rules that guide the process flow, i.e., a process model that an engine can interpret and use to guide the process execution. Informal processes, that are complex and constantly changing, are no fit for model-based approaches. Moreover, workflow models usually assign a given activity to an executor, hindering collaboration among the stakeholders.

[1] http://fenixedu.org

3.1. USER EXPERIENCE

One of the most important aspects when introducing a new information system is the user experience. Since users are already familiarized with e-mail, we tried to be extremely careful while disrupting that communication experience. A great example of such disruption is related to the request-response synchronization paradigm that is so built-in into the user experience. Such paradigm is also one of the main limitations that we listed before: the number of copies and communication channels exponentially increases with the number of participants (for N participants in an e-mail thread, there are ((N-1)*N)/2 possible communication channels). Hence, we need to drop the request-response paradigm, and inherit the multi-participant collaborative aspect. Although we dropped support for request-response interaction pattern, we kept a User Interface (UI) similar to that of e-mail folders (see Fig. 1).

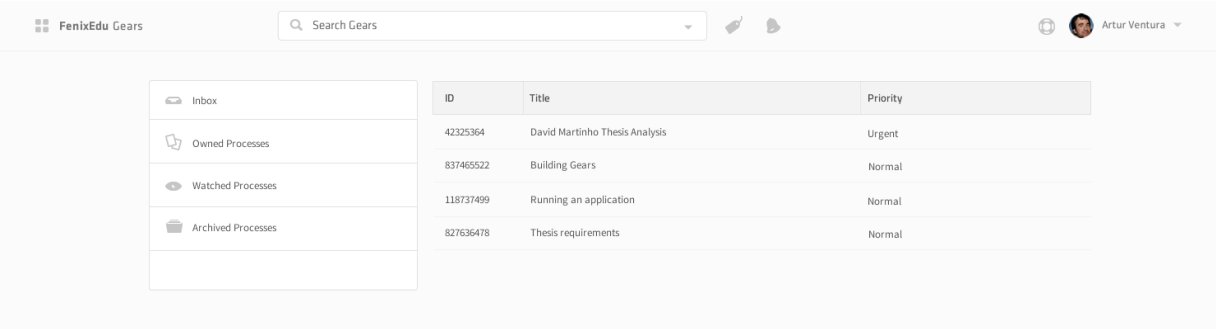

Fig. 1: Folders View: Inbox

While in the Folders' View, users can access their Inbox folder, where all new process instances, for which they are invited to participate on, are listed. Process instances created or managed by the user are listed in the Owned Processes folder. Also, users can choose to follow specific processes by watching them. Such processes will be listed in the Watched Processes folder. Finally, when the processes where the user participates are marked as being concluded, they will only be listed in the Archived Processes folder. In addition to these built-in folders, the user can create “smart folders”, which essentially are filters that list all the processes that match a particular set of tags, specified for each of such folders. For example, if the user wants to list all the processes related to Master Theses submitted in 2015, it can create a “smart folder” that matches the tags “master-thesis” and “2015”, assuming that such tags were used to categorize all the processes related to Master Theses submitted in 2015.

3.2. PARTICIPATION-ORIENTED

During our case study, we identified several key interactions that would drive the process execution progress. Such interactions have two or more participants who communicate and may then trigger new interactions with other users. In GEARS, any such interaction is designated by participation. All the actors of any given participation can exchange comments and files.

3.3. PROCESS VIEWS

In GEARS, there are four main process views: the Participation Feed, the Timeline View, the Process Event Stream View, and the Process Files View.

PARTICIPATION FEED

The Participation Feed lists all the active participations that are relevant for the user, ordered by relevance. The relevance is based on the timestamp of the last interaction between the users and the participations. When a user performs some action in a participation, for instance, posts a comment, the relevance value for each participant, other than the user, is updated to the current timestamp. In the Participation Feed, participations are ordered by decreasing order of their relevance value.

TIMELINE VIEW

In the Timeline View the user can see all entries from a given process, such as participations and events, from the most recent one to the oldest one. Fig. 2 depicts an example of the Timeline View of a process. In the example, there are 5 different participations within the timeline. Each participation has four main perspectives: the discussion perspective, the participants perspective, the files perspective, and the summary perspective. The discussion perspective contains a list of all the comments made by the participants, along with the stream of actions such as file uploads and their updates. The participant perspective shows the list of current participants, and allows any participant to invite more participants to the participation. Moreover, the files perspective lists all the participation files, along with their previous versions. Finally, the summary perspective displays an optional statement that summarizes the participation outcome when it ends.

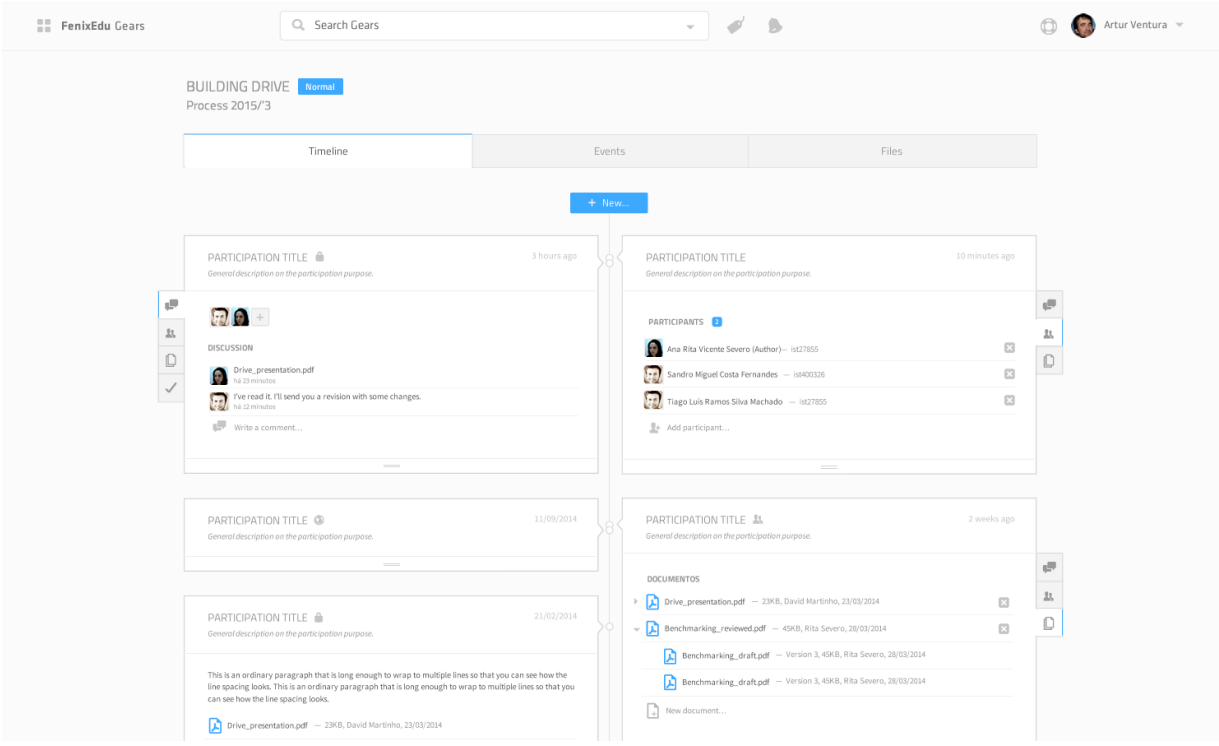

Fig. 2: Example of the Process Timeline View

PROCESS ACTION STREAM VIEW

The Process Event Stream View lists all the action events executed within the process by its participants (e.g. participation creation, file uploads and updates, new comments, etc). The entries of this stream are ordered chronologically. This view is particularly important for participants to acknowledge the latest actions that took place on the process instance.

PROCESS FILE VIEW

To enhance the efficiency of finding a file within a process, instead of checking each participation, GEARS offers a Process Files View that lists all files that can be accessed by the user within that process. With the Process Files View, the user can lookup and access process files much faster than by using e-mail, where he would need to check all the e-mails and their respective attachments. Additionally, the user would need to check whether a given e-mail contains in fact the latest version of the necessary attachment. GEARS offers a list of all the latest versions of each file, allowing the user to access older versions if necessary (see Fig. 3).

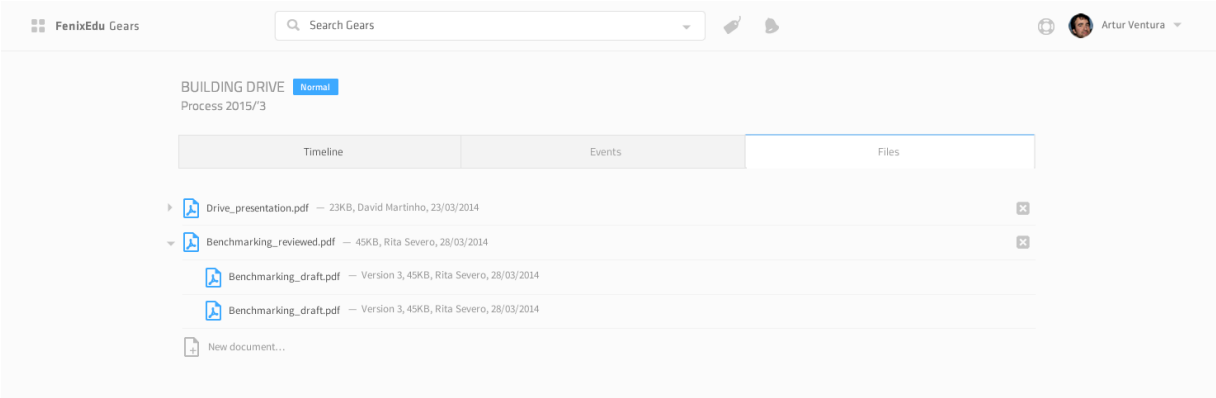

Fig. 3: Example of the Process Files View

3.4. INTERACTION PATTERNS

In GEARS, the most granular interaction context is called a participation. Participations exist within processes, and consist on a focused discussion between two or more participants. Within a participation, the participants can essentially write comments and upload files. When such interaction is finished, any of the users can propose a public summary, which represents the outcome of such participation. A participation is like a virtual meeting between several participants, where they discuss one or many subject and can, optionally, make a public summary (statement and files summarizing the conclusions of the participation) available to other people involved in the process but not necessarily involved in that specific participation.

PARTICIPATION PRIVACY POLICIES

Each participation is created according to a particular visibility policy: participations can be hidden, private or public. Hidden participations are only visible to their participants, meaning that other process participants do not even know of their existence. If the optional summary is proposed within the hidden participation, only its participants can view that summary.

Private participations are known to all process participants, but only the participants of the participation can access its content (participants, files and comments). In the end of a private participation, any of the participants can propose a public summary that is voted by all the participants. If the summary is approved by all the participants, it is made available to the remaining process participants. Until then, all other process participants can only see the private participation's title.

Public participations are participations where everyone participating in the process can see its content: title, participants, comments and files. However, only the participation participants are allowed to interact within the participation (i.e. make comments and upload files). Although public, if the participation is too long, the participants might optionally create a summary that summarizes the outcome of that participation. It is more efficient to check the summary than to read the whole participation and cherry-pick its real outcome.

PROCESS EVENTS

Sometimes, in specific processes, there are events (dates) that are relevant to the process development and should to be remembered. For example, in the Master Thesis process, the defense is an important event, therefore its date should be clearly presented in the process given its relevance both for the jury and the candidate. Hence, besides participations, GEARS allows any process owner to also create process events within the process timeline (Fig. 4). These events display a title, a small description, date and time of the event, as well as the place where it will occur. All process participants may manifest their attending intention by stating if they are Going, Maybe Going or Not Going.

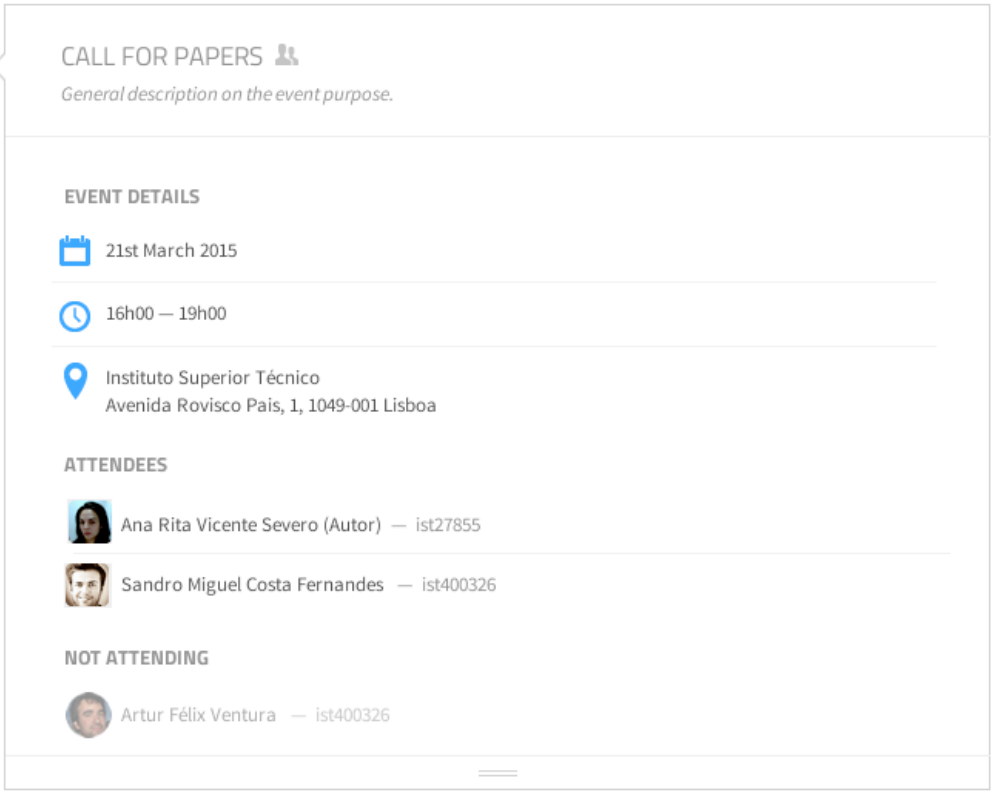

Fig. 4: Example of the a Process Event Entry.

PARTICIPATION FEED

When participating in multiple processes, participants become less productive if they have to open the Process Timeline view and search for participations of interest that need immediate action. To address this problem, GEARS offers a Participation Feed view where all participations of interest, despite their process, are listed. Although the participation feed lists participations from different processes, the process context (title and ID) is always included so that the user can have context of execution. Using this view, users are able to perform actions on their different participations without having to jump among process timelines.

4. IMPLEMENTATION

Currently, the GEARS project is about one year long, and its implementation status just reached the first release candidate. In a near future, GEARS will be released as an open-source sub-project under the FenixEdu project umbrella.

4.1. DOMAIN MODEL

As mentioned before, GEARS allows users to create new business process instances, in which they can create new events, and interact with other users within participations. In Fig. 5, there is a simplification of the GEARS Domain Model.

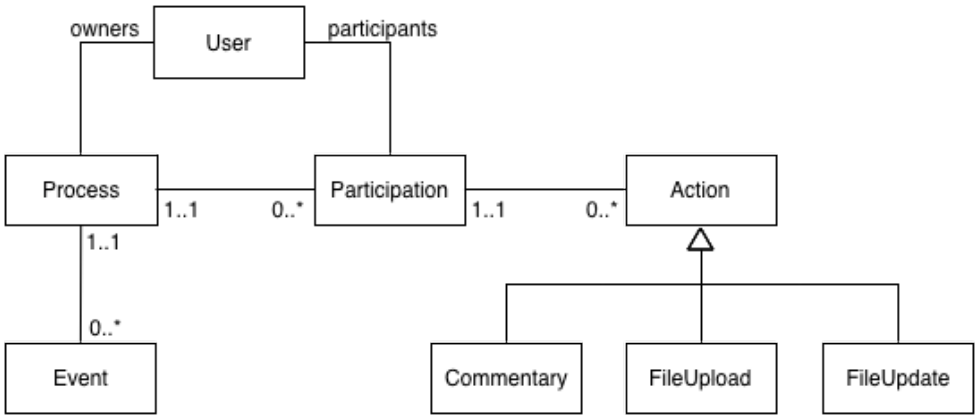

Fig. 5: GEARS Domain Model

Process owners are allowed to create new events and new participations. Events consist of information like the event's name, place and date. Such information is accessible for all process participants, i.e. to process owners and anyone that is member of at least one participation. Within a participation, any of its participants can execute one of three main actions: upload a new file, update an existing file, and write a comment. Participants can also invite additional participants to their participations, but such action is not logged within the action class.

This model is simplified for the sake of simplicity, and to focus on the main underlying concepts of the GEARS system. There are more details in the model, such as the public response, which is not relevant to understand the main entities that compose the GEARS system.

4.2. TECHNOLOGY

The GEARS system is implemented in both Javascript and Java. More specifically, the client-side is fully implemented in Javascript, using AngularJS[2] framework, while the server-side is developed in Java, using the BennuFramework[3] which is a web framework developed and maintained by FenixEdu[4] team. In an architectural perspective, the client-side consumes a REST API that implements the main business logic of the GEARS application.

This project is composed by two main modules: one is the Core module where all the REST API endpoints that handle all the business logic are implemented; the other one is the UI module where all the frontend is implemented (see Fig. 6).

The application's data is stored in a MySQL database. FenixFramework[5] is used because it offers an abstraction layer around the persistence mechanism, in such a way that developers handle domain's persistence using usual Java classes instead of having to deal with the database itself.

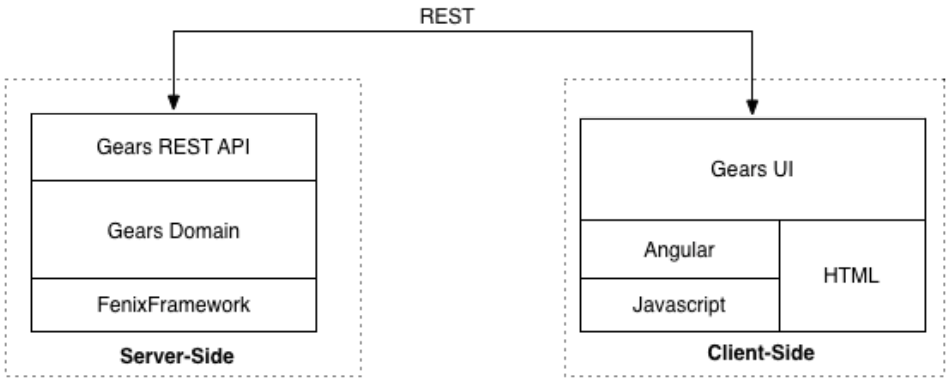

Fig. 6: GEARS Architecture

Also, during build-time, Browserify[6] is used in the client-side to allow frontend Javascript code that uses require statements, similarly to NodeJS[7] modules. To manage third-party frontend dependencies, such as AngularJS, Bootstrap and others, we use Bower. To integrate all these tools in our build process we created a maven plugin for that purpose. Such plugin essentially downloads all the required artifacts such as NodeJS, Bower[8], and other NodeJS libraries, and configures them easily to help build a more efficient development environment, as well as preparing the code for production (minification of Javascript for example).

---

[2] http://angularjs.org

[3] http://github.com/FenixEdu/bennu

[4] http://fenixedu.org

[5] https://github.com/fenix-framework/fenix-framework/

[6] http://browserify.org

[7] http://nodejs.org

[8] http://bower.io

5. CONCLUSIONS AND FUTURE WORK

This paper presents GEARS, a new approach to ad-hoc workflow execution that focuses on providing the traceability that e-mail messaging fails to provide. In e-mail, the same business process execution is scattered among different threads with different subjects, and their attachments are replicated in different messages. GEARS provides a single point of communication to each process instance, allowing people to interact seamlessly. In such interactions, users can share their insights by writing comments or uploading files (for which versioning is supported). Each process instance is viewed in a timeline format, that lists the participations in chronological order, as well as events of interest to the process instance participants.

The application will be deployed in Técnico Lisboa, to a community of 15000 users, by the end of May 2015. We intend to capture usage statistics for both e-mail and GEARS in order to understand to what extent are people migrating from e-mail to GEARS.

6. REFERENCES

[1] Whittaker, Steve, and Candace Sidner. "Email overload: exploring personal information management of email." Proceedings of the SIGCHI conference on Human factors in computing systems. ACM, 1996.

[2] van der Aalst, Wil MP, and Andriy Nikolov. "Mining e-mail messages: Uncovering interaction patterns and processes using e-mail logs." International Journal of Intelligent Information Technologies (IJIIT) 4.3 (2008): 27-45.

7. AUTHORS' BIOGRAPHIES

Luis Guerra e Silva - pt.linkedin.com/pub/luis-guerra-e-silva/0/b20/a30/en

Luis Cruz - Luis Cruz is an open source enthusiast, with contributions mainly focussing on academic information systems for higher education. He likes to indulge in software architecture, design patterns and application performance profiling and analysis. He is currently a principal contributor to the FenixEdu project and is the coordinator of the Area of Applications and Information Systems at Instituto Superior Técnico. He has a degree in Information Systems and Computer Engineering from Instituto Superior Técnico.

Samuel Coelho - pt.linkedin.com/pub/samuel-coelho/

Rita Severo - pt.linkedin.com/pub/rita-severo/39/b08/a08/en

Artur Ventura - pt.linkedin.com/pub/artur-ventura/14/3/485/en

David Martinho - pt.linkedin.com/in/davidmartinho

João Carvalho - pt.linkedin.com/pub/joão-pedro-carvalho/43/665/b68/en

Ricardo Barata - pt.linkedin.com/in/rsbarata

Pedro Santos - pt.linkedin.com/in/pedromrsantos/en