Reliable Document-centric Processing in a Loosely Coupled Email-based System

Magdalena Godlewska
University of Gdansk
Faculty of Mathematics, Physics and Informatics
Gdansk, Poland
Email: maggod@inf.ug.edu.pl

***Abstract*—Email is a simple way to exchange digital documents of any kind. The Mobile INteractive Document architecture (MIND) enables self-coordination and self-steering of document agent systems based on commonly available email services. In this paper, a mechanism for providing integrity and reliability of such an email based agent system is proposed to cope with message soft or hard bounces, user interrupts, and other unexpected events. This mechanism consists of a system acting as a "ground control" for migrating documents and a set of protocols that improve the implementation of document coordination patterns. It allows for an estimation of the global state of a distributed loosely coupled agent system and making top-down decisions in unforeseen situations.**

Keywords–multi-agent systems; collaborative work; electronic documents; email-based systems

I. Introduction

A knowledge-based organization is a management idea, describing an organization in which cooperating people use knowledge resources to achieve organizational goals. People are the key intellectual resource but only collaboration with other workers in accordance with the organizational procedures enables converting knowledge of individuals to knowledge of organization [1].

Knowledge workers communicate through the exchange of documents constituting units of information. Nowadays, email has a dominant position in the computer mediated communication and document exchange in the workplace [2]. Email messaging provides an easy to use simple textual form and allows to disseminate attachments in any format to one or multiple recipients.

The MIND architecture [3] is a proposition of a document-centric uniform interface to provide both effective communication of content and coordination of activities performed on documents. MIND is a solution that augments email messaging with proactive documents, capable of initiating process activities, interacting with individual workers on their personal devices and migrating on their own between collaborators. Thus, each MIND document is a mobile agent. Document-agents have built-in migration policy to control their own work-flow and services to proper processing contained information. Section II contains a more detailed overview of the MIND architecture.

The migration path of the document-agent contains all information and status of the workflow process to perform it locally on users' devices. An email client installed on each worker's device participating in the process needs to be extended with functionality to activate the document-agents and switch documents between the activity and transition phases of the workflow. This special email client with workflow enactment capability has been implemented as a Local Workflow Engine (LWE) [4]. All LWEs participating in the process and performing independently form together both a loosely-coupled agent system and a distributed workflow enactment service. Section III outlines generic functionality of LWE and the idea of distributed workflow enactment service.

In the LWE-based MIND system, individual knowledge workers perform activities on documents independently, using their personal devices, and yet collaborate on achieving a common goal. This is possible owing to the migration policy embedded in each document. This policy defines for each document a workflow process composed of specific document-flow patterns that provide process wide coordination. The document-flow patterns [4] are the result of analysis of the coordination patterns proposed by van der Aaalst [5] under the assumption that email is the transport layer for document migration. The work of van der Aaalst shows that a relatively small and well defined set of collaboration patterns contains building blocks of arbitrary complex workflow processes in real organizations. Thus, the document-flow patterns that directly follow the collaborations patterns for the proposed MIND architecture enable modeling and coordination of any workflow process.

The crucial services for MIND implementation are executability and mobility. The former involves activating document-agents to enable their autonomous execution, while the latter involves transporting them between users' devices in accordance with the migration policy. These services have already been implemented and described before [3][4]. In the prototype system, mobility has been implemented based on email as the transport layer.

In most cases, these mentioned services are sufficient to properly perform the MIND agent system. However, in human organizations, some situations unforeseen by the designer of the process may occur and the reliable system should be able to cope with them. For a distributed loosely coupled and interactive system it is impossible to determine a global state of all document-agents. Consequently, it is not possible to determine where all documents are located at a specific time. A document may be lost and it often remains beyond the knowledge of the authors, who have edited it earlier. There are various reasons why documents may be lost: a problem with a transport layer, an error of a local environment or an unexpected user behavior. The initial workflow process definition makes it possible to search for documents in the specified places. However, the process may be modified during its execution. Therefore, a document may take a path that was not originally designed and a document originator has

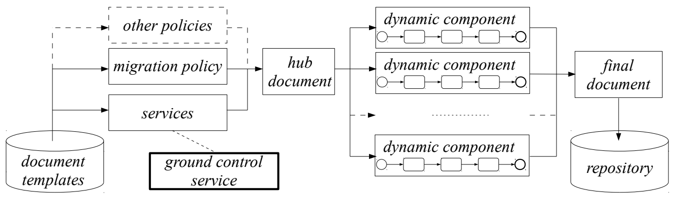


Figure 1. The MIND document lifecycle

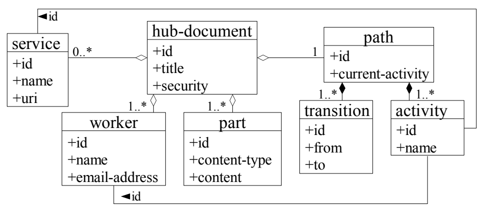


Figure 2. Dynamic form of the MIND architecture [4]

not information about its definition.

Thus, the *reliability* of the MIND agent system is a service that makes the system more useful and trustworthy than the typical email-based communication. It allows for an estimation of the global state of a distributed loosely coupled system, taking into account transport layer errors, unforeseen actions of users and process modifications.

Thus, Section IV identifies problems associated with distributed workflow execution. Section IV-A focuses on the problems associated with email as the transport layer for the MIND documents, while Section IV-B presents problems that may occur in specific document-flow patterns. In particular, it is interesting the canceling pattern due to the loosely-coupling principle at operating of the MIND agent system.

Section V presents a concept of a ”ground control” service which introduces the ability to track document-agents globally and solve some of the problems associated with the documents flow. The service is designed to receive signals containing the status of the document from the LWE clients and send control signals to LWEs that resolve situations incompatible with the designed workflow. The proposed syntax of a notification sending to the ”ground control” service is adapted to document-flow patterns. Further, this section outlines two pilot implementations of the ”ground control” service – one using the Handle System [6], and another based on an email-based notification system.

Section VI surveys previous work related to a document-centric processing and a reliability of workflow enactment in distributed loosely-coupled systems.

II. The MIND architecture

The MIND architecture enables the new agent-based distributed processing model. Traditionally, electronic documents have been static objects downloaded from a server or sent by email. MIND allows static documents to be converted into a set of dynamic components that can migrate between collaborative workers according to their migration policy.

The concept of the MIND document life cycle is illustrated in Figure 1. At the beginning of the knowledge process, some originator forms a *hub document* based on document templates that includes migration policy, which specifies the steps of the process and services that will be performed on different parts of the document during the process. The hub document is changing to mobile components that meet their mission in the distributed agent system. Each component performs its migration policy and interacts with workers of the organization.

The MIND architecture makes possible a radical shift from *data-centric* distributed systems, with hard-coded functionality, to flexible *document-centric* ones, where specialized functionality is embedded in migrating document components and some generic or supporting services are provided by local devices or external servers. The essence of the MIND architecture is that the documents have capability of self-organization and self-steering during the process execution.

Figure 2 outlines the dynamic form of the MIND architecture. It includes five components: *hub-document* is the main component and it contains basic information about the document, *worker* component contains data about workers who participate in the process, *part* component defines parts of the document, *service* component contains information about services that can be performed on different parts of the document during the process, and *path* component defines migration policy of each part of the document. It specifies the steps of the process and activities that should be performed at each step of the process.

The service objects provide document functionality that makes it proactive. Three types of services are possible: *embedded* that are transferred together with the document, *local*, which may be acquired by the document components from local worker's device, and *external*, called on the remote hosts by the worker's system at the request of arriving document.

III. Distributed workflow enactment

A key feature of the MIND architecture is physical distribution of business process activities, performed dynamically on a system of independent personal devices. MIND documents have built-in process definition and functionality (the respective path and embedding service components mentioned in the previous section). This makes them agents, which are autonomous and mobile. Especially, they are independent of any particular platform supporting workflow enactment and they are capable of launching individual activities onto various workers' devices which maintain process coordination across the organization.

Workflow enactment service interprets the process description and control sequencing of activities through one or more cooperating workflow engines [7]. Even if the workflow engines are distributed, workflow enactment is centralized in most of the implementations, because the control data must be available for all engines. In the MIND architecture, all data needed for workflow enactment are embedded in documents [4]. This allows for implementation of distributed workflow enactment service consisting of LWEs.

The idea of the distributed workflow enactment service built on top of LWE clients and email transport layer is illustrated in Figure 3. In the prototype system, LWE was implemented as lightweight email client installed on personal devices of each worker. Each LWE is independent of other LWEs, so it can be implemented in any technology and adapted

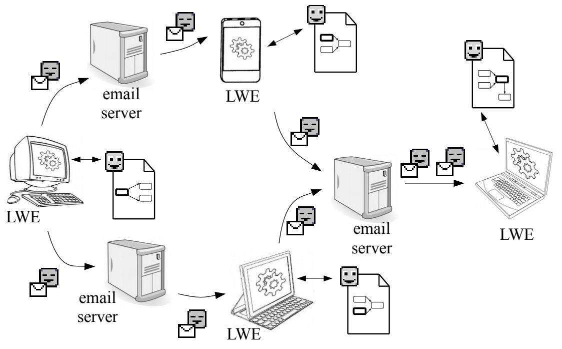


Figure 3. Distributed workflow enactment service based on LWE clients and email transport layer (LWEs are symbolized as gear wheels).

to requirements of particular devices, especially mobile devices such as tablets and smartphones. Also, it may be implemented as a plug-in to existing email clients.

States of the LWE correspond to the phases of a document lifetime and the initial state is when a message with a document is received, i.e., noticed by LWE in the worker's mailbox. The LWE downloads the document on the local device and activates it, which means launching its embedded functionality. The activated document may interact with the knowledge worker, his/her local system and some external services. The interaction begins with obtaining the document path component and determining the current activity that should be performed in this particular step of the process. If the next activity is intended for another worker, the document is serialized, packed and sent as an attachment to the next worker's email address.

LWE is capable of recognizing and executing all document-flow patterns contained in the path component. More details about the document-flow patterns are in Section IV-B, which presents a discussion about their execution in a loosely-coupled distributed system consisting of the LWE clients.

IV. Problems of reliable workflow execution

MIND and LWE clients form a distributed workflow enactment system in which the coordination of activities is based on control data contained in the documents. In most cases, it ensures that the documents arrive at a specific location at a specific time. Nevertheless, some situations unforeseen by the designer of the process may occur in loosely-coupled system.

First of all, the document may be lost: during the transfer by email, due to failure of the local system, accidentally deleted by the user. The transfer of the document may also be delayed to miss the designed deadline. The knowledge worker may also make a decision unforeseen by the workflow, e.g., cancel some document flow or modify the workflow path, which is just not possible in typical message passing via email.

Figure 4 shows the path of the document from a sender to a recipient and indicates points where some problems may occur. Points ②–④ are associated with several well known email transport layer problems briefly described in Section IV-A, while points ① and ⑤ indicate problems with document-flow patterns execution by LWE clients, detailed in Section IV-B.

A. Email transport layer problems

Email message is a simple textual form combined with attachments in any format. It can be sent to one or multiple

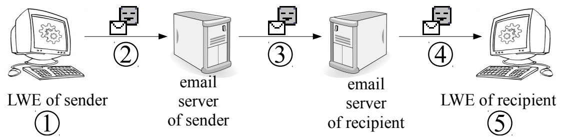


Figure 4. LWE to LWE connection. The numbers indicate points, where some problems with the transport of the document may occur.

recipients and supports asynchronous work. Email mechanisms have a reputation of being robust and trustworthy since its invention a few decades ago, as email messages reach their recipients in most cases without problems. Nevertheless, there is a list of problems associated with the delivery of messages.

The first step in the email processing model is to submit email message by an email client (Mail User Agent – MUA) to a sender Simple Mail Transfer Protocol (SMTP) server (Mail Transfer Agent – MTA) [8]. Figure 4 indicates it as point ②. This step may fail due to the lack of network connection, incorrect SMTP server configuration or SMTP server failure. The message usually remains in the sender outbox and the email client tries to send it again. Configuration of SMTP server for LWE client does not differ from the configuration of other email clients and does not require any special functionality. Temporary lack of network connection is a typical situation for mobile devices. SMTP server failure is a rather transient situation that can be solved by resending the email message.

In the next step, sender MTA transfers messages to the receiver MTA mostly by SMTP protocol (point ③ in Figure 4). SMTP server should deliver the message or notify about any problem [8]. The SMTP reply consists of a three digit number often followed by some text for the human user. The message may be rejected, however, in a transient or permanent way. In transient situations, the SMTP client should try to send the message again. In the case of permanent errors, the SMTP client should not repeat the exact request. After a failed attempt to send a message, the sender SMTP agent sends a notification message to the mail user agent. This notification message is known as a Delivery Status Notification (DSN) or email bounce [9].

Nevertheless, receiving of email bounces does not necessarily mean that the message has not been delivered and, conversely, the lack of notice does not necessarily mean that a message has arrived to the recipient. For instance, the receiver SMTP server may silently drop message to protect themselves from attacks [8]. Many SMTP servers are configured to block messages categorized as spam based on DNS blacklists or anti-spam filters [10].

Receiving a message by the SMTP server and placing it in a user's mailbox does not imply that the user will read it. Point ④ in Figure 4 indicates the problem of the recipient's email server – email client communication. Firstly, some messages may be marked as spam and placed in the spam folder in user's mailbox. In this case, the frameworks to build mail applications (like Java Mail [11] and IMAP – Internet Message Access Protocol [12] used in the LWE implementation) often enable access to the spam folder. In fact, also the email client may have its own spam filters and other solutions to manage received messages automatically, like the automatic responses software (e.g., ”out of office” message) [13].

Next to email bounces and automatic responses there is one more type of notifications, the Message Disposition Notifications (MDNs) [14]. These notifications are intended to report of the disposition of a message after it successfully reaches a recipient's mailbox. The MDN can be used to notify the sender of any of several conditions that may occur after successful delivery such as display, printing or deletion of the message. Allow mail user agents to keep track of the message (only) in its subsequent step of the flow. The sending of the response depends on the functionality of recipient email client and often on the decision of the recipient.

Message tracking is also possible through email tracking services like ReadNotify [15] or WhoReadMe [16]. These services add to the message some hidden information: picture, or pieces of HyperText Markup Language (HTML) code (like IFRAMEs). Tracking is hidden from the recipient and not too elegant.

There is yet another reason for which the message may not reach the mail user agent - the human action. The recipient may accidentally or intentionally delete the message from his/her mailbox, move it to a different folder or mark it as a spam. Also, his/her email client or a local system may fail.

B. Document-flow patterns execution

In mailing systems, notification mechanisms can provide the status of messages in their the next step of flow, but never any further. It can be said that the email message can store history of its own flow, inform about its next step, but does not "know" its future flow. The MIND document has an embedded workflow path, thus it has information about whole its flow and about flow of other documents in the process. However, a worker which finished his activity has no control on further flow of document – this knowledge is built in document, which has left his device.

In some cases, the location of the document may be required for the proper execution of the workflow process, especially in unexpected situations, like a lost document. LWE temporarily stores copy of documents in the worker's mailbox, in case the process has to be recreated from a certain place. Searching for a document in all places indicated by a workflow is possible but often time-consuming and costly, and may not take into account the modification of the path during the process execution.

This paper proposes a "ground control" external service for receiving and storing notifications from LWEs about status of documents. Each notification from LWEs contains information about: process id, document id, current activity id, and sender of the notification. This section presents what other information about the document should be included in the notifications for reliable coordination of all document-flow patterns.

Based on the work of van der Aaalst [5] and the result of previous research [4], three categories of document-flow patters have been identified: distributed state patterns, coupled state patterns, and embedded state patterns.

1) Distributed state patterns: These patterns describe situations in which the next activity or activities can be determined solely on the state of the current activity. Four patterns of this type have been distinguished: sequencer, splitter, merger and iterator.

a) Document sequencer: This pattern involves a knowledge worker sending a document to another worker. The document may be sent in its entirety in one message or it may be partitioned into several messages. In this basic situation, the following problems may occur: a sender may receive bounce notifications from each sent message and recipient may not receive all messages. However, a bounce notification does not always mean that the recipient has not received the message in a timely manner. In this pattern, the notification should contain one of the three route-status: sent (sends from sender's LWE after sending the document), received (sends from recipient's LWE - after receiving the document) or bounced (sends from sender's LWE - after receiving the bounce notification). It is possible, that some notification does not reach to the "ground control" service or arrives in the wrong order. Thus, in all patterns, the *received* status and the subsequent *sent* status are considered to more important then previous *bounced* and *received*.

b) Document splitter: This pattern creates identical copies of the document or partitions it into separate fragments. The resulting documents are next sent to the respective knowledge workers specified in the migration policy. These documents, either copies or fragments, get new document IDs. The parent document is considered to be delivered if all its child documents have been delivered. Thus, the *sent* route-status is given to each parent and child documents. The parent document has also assigned a *splitted* document-status and references to the child documents are indicated. Each arrived child document gets the *received* status individually. Once all the child documents have the *received* status (or the subsequent *sent* status), the "ground control" service gives automatically the *received* status to the parent document. The child documents are determined by the references. The *bounced* status is also assigned to each child document separately.

c) Document merger: This pattern complementing the document splitter pattern merges all received documents in one. Of course, this pattern may involve various document functionality, depending on whether the preceding splitter has been cloning or decomposing. But before merging, all the expected documents must be delivered. The LWE client on the basis of path component of the first received document determines the number of expected documents that have to be merged. Each of the arrived document gets the *received* status. When all documents are collected, they are merged and a new document gets the *received* route-status and constitutes documents get the *merged* document-status and reference to this new merged document. The document merger fails when at least one child document has been not received. In exceptional situations, decision about completing merger before receiving all child document may be made.

d) Document iterator: This pattern enables repeated execution of some sequence of activities controlled by a condition specified in the respective document migration policy. The route-status is assigned as in document sequencer, but the activities can be performed several times and notification may be received by "ground control" service in incorrect order. Thus, the activity id and route-status is not enough to determine where the document resides. To solve this problem, some basic partial ordering mechanism, like Lamport's timestamps [17], has been used. The path component of the document has a timestamp attribute that is incremented by LWE. When the documents are merged, the new one gets a maximum value of all merged documents' timestamps plus 1. Thus, each notification contains also a timestamp value.

2) Coupled state patterns: Sometimes completion of an activity performed by one worker may require a notification on a state of some activity performed by another worker somewhere in the organization. That involves the notion of asynchronous signals, sent between different parts of the workflow process. Three document-flow patterns of this kind have been distinguished: deferred choice, milestone and cancel activity.

a) Deferred choice and milestone: These patterns are used to deal with situations when the current activity of one worker has to be blocked until a signal notifying on some external event has been received from another worker. Both patterns require a proactive document to provide a worker's device with a semaphore and embedded functionality to handle it. Initial value of the semaphore is closed, so if the signal from another worker has not been received, the current activity is blocked. Upon receiving a signal, the waiting activity is resumed. Deferred choice is used when sending a given document has to be postponed until the worker gets information to whom it should be sent. Milestone just blocks some activity of one worker by another. The problem appears, if the signal does not arrive within the specified time and the received document activity can not be proceed. In this case, the route-status of the document is *received* but the signal-status is *waiting*.

b) Cancelling pattern: Implementation of this pattern depends on what exactly should be cancelled. If a particular activity should be cancelled, a cancellation signal is sent only to the LWE client responsible for its performance. The decision on canceling the activity is immediate for the receiving device or does not make sense any more if the document has been sent to another worker.

More problematic situation is to cancel the document, because it requires the designation of its location. It is possible to search for a document in all places indicated by the workflow, but the "ground control" service can significantly reduce this set of places. If the route-status of document is *received*, the cancellation signal is sent only to the sender of that notification. After a successful cancellation, LWE sends the *cancelled* route-status.

If instead the *cancelled* route-status, the "ground control" service receives the *sent* status, it can start chasing the document. This situation is shown in Figure 5. To increase the chance of success, a cancellation signal is sent to, say, three subsequent activities for each possible path of the document flow. The three cases are possible for each activity: an activity was finished, an activity is currently being preformed or waiting for a document. Figure 5a) shows successful cancellation, i.e., the "ground control" service received a *cancelled* notification from all possible paths of the document. Figure 5b) shows cancellation potentially successful but not yet completed. While Figure 5c) shows the failed cancellation - the cancelling process should be continued for the subsequent activities on this particular path.

It is worth mentioning that the rate of the document flow is measured in minutes or hours, even days, rather than seconds. For example, the Intel's Email Service Level Agreement defines the acceptable time frame for replying to emails to 24 hours [18]. Thus, chasing the document will not be so much demanding as it might appear to be.

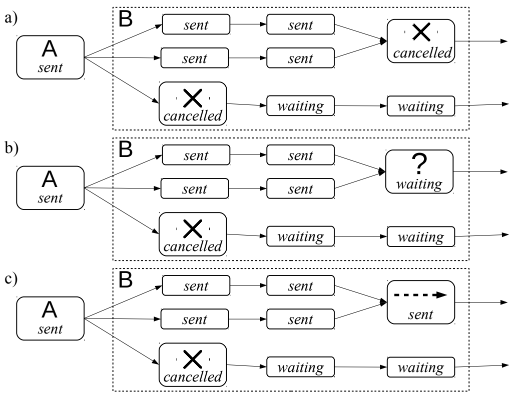


Figure 5. Cancellation of the document

3) Embedded state patterns: Performing an activity by some worker may require a subprocess delegated to someone else, with activities not specified originally in the migration policy of the arriving document. States of such a subprocess are embedded in the state of the current activity enabling that.

a) Internal subprocess: If the current worker is authorized to extend the original migration policy of a document with new activities, they constitute an internal subprocess. Neither the structure of the internal subflow nor identity of added workers have to be known earlier to the workflow designer. The notification from the subflow activities are the same like from other activities, but the "ground control" service has only the structure of the designed workflow. Thus, for reliable coordination of subflow, its structure and identity of added workers must be sent to the "ground control" service. If the "ground control" does not have the current data of the subprocesses, tracing a document, and in particular, the cancellation may not be possible.

b) External subprocess: The performed activity may call some external subprocess, which structure are unknown for both, workflow designer and the performer of the current activity. The external subprocess is often performed outside of the organization, thus, it is not traced by the "ground control" service. Only the lack of received notification at the end of the subprocess within the specified time may indicate troubles.

The document-flow patterns analysis allowed for formulating the syntax of notifications. The route-status type should be one of the: sent, received, bounced, cancelled and finished, the optional document-status can be one of the: splitted or merged. The signal-status can be waiting or just indefinite. Each notification contains also timestamp value. LWE performed a first activity adds to the notification a migration path and information about workers. If it notices a modification of the path component by adding a subprocess, it also sends definition of subprocess and information about new workers to provide the most recent data of the process. Table I summarizes the problems associated with the reliable execution of presented patterns.

V. GROUND CONTROL SERVICE

This is an external service intended for a central document tracing to ensure the reliability of distributed workflow

TABLE I. RELIABILITY OF PATTERNS EXECUTION

PATTERN	PROBLEMS TO SOLVE
Sequencer	Check whether the document has reached the recipient. Route-status: sent, received, bounced.
Splitter	Check whether all constituents of the splitted document have reached the recipients. Route-status: sent, received, bounced. Document-status: splitted (for splitted document + references to constituents).
Merger	Check whether all documents that should be merged into one have reached the recipient. Route-status: sent, received, bounced. Document-status: merged (for merged documents + references to new document).
Iterator	Check whether the document has reached the recipient as many times as it has been established in the loop. Route-status: sent, received, bounced. Timestamp to determine the order of the activities.
Deffered choice Milestone	Check whether both the document and the signal have reached the recipient. Route-status: sent, received, bounced. Signal-status: waiting (or indefinite).
Cancelling	Check whether the document has been cancelled. Route-status: cancelled (when it succeeded).
Internal subprocess	Track a subprocess added during the workflow process execution. Attach subprocess sources to the notification.
External subprocess	This pattern is not tracked by the ”ground control” service.

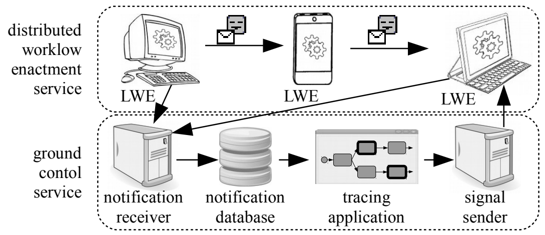


Figure 6. The *Ground control* service architecture

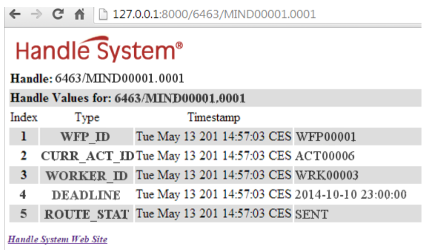
execution. The workflow enactment remains distributed and may be still performed without it, however. The intention of the ”ground control” service is to collect notifications from LWEs in order to determine the approximate global state of the distributed document flow and to make top-down decisions in some unforeseen cases. The document policy component decides whether the notification has to be sent or not.

Figure 6 presents the concept of this service. It constitutes a notification receiver, i.e., a service receiving notifications from the LWEs via the particular transport layer. Then, the notifications are parsed and placed in the database. The notification database has some functionality, e.g., trigger that gives automatically the *received* status to the splitted document, after all its child documents have also got this status.

A tracing application visualizes the workflow process and marks the currently executing activities, designated on the basis of the notifications. It is also an interface for some users allows for monitoring the process and/or makes some top-down decisions. Some decisions may require sending the notification signal to the particular LWE. Signals are generated by the tracing application and transfered by the signal sender service.

A. Implementation

Two possible implementations of transferring and storing notifications were taken into account. The former uses the Handle System, while the latter uses an email-based notification system.


127.0.0.1:8000/6463/MIND00001.0001

Handle System®

Handle: 6463/MIND00001.0001

Handle Values for: 6463/MIND00001.0001

Index	Type	Timestamp	
1	WFP_ID	Tue May 13 201 14:57:03 CES	WFP00001
2	CURR_ACT_ID	Tue May 13 201 14:57:03 CES	ACT00006
3	WORKER_ID	Tue May 13 201 14:57:03 CES	WRK00003
4	DEADLINE	Tue May 13 201 14:57:03 CES	2014-10-10 23:00:00
5	ROUTE_STAT	Tue May 13 201 14:57:03 CES	SENT

Handle System Web Site

Figure 7. A handle for the MIND document

1) Handle System [6]: is a solution for assigning, managing, storing and resolving persistent identifiers for digital objects on the Internet. It includes a set of protocols enabling a distributed computer system to store identifiers of digital resources and resolve those identifiers into the information necessary to locate and access the resources. This information can be changed to reflect the current state of the identified resource without changing the identifier. The most popular system based on Handle System is DOI (Digital Object Identifier) [19] used for persistent citations in scholarly materials, research datasets or European Union official publications.

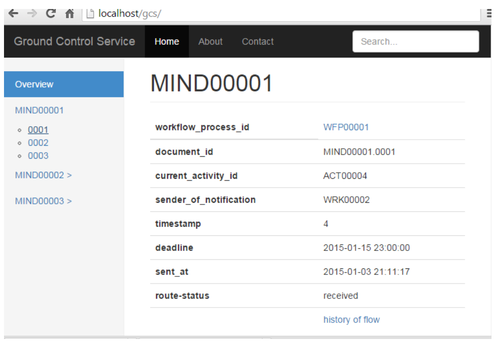
The Handle System defines a hierarchical service model. The top level consists of a single handle service, known as the Global Handle Registry. The lower level consists of all other handle services, known as Local Handle Services. The Handle System provides the Java-based Handle Server and a set of tools needed for the Local Handle Service installation. The Global Handle Registry is used to manage any handle namespace and provides the service used to manage naming authorities. The Local Handle Service and its responsible set of local handle namespaces must be registered with the Global Handle Registry and gets a unique prefix.

The Handle System provides unique persistent identifiers called handles for digital objects, such as the MIND document. The handle is a character string that consists of two parts: its naming authority and a local name separated by the ASCII (American Standard Code for Information Interchange) character ”/”. Each handle may have a set of values assigned to it. A handle value may be thought as a record that consists of a group of data fields. Every handle value must have a data type. The Handle System predefines a set of data types and allows for defining another.

Thus, the ”ground control” service can use the Handle System to create an unique handle for each migrating document (see Figure 7). Each handle has a set of values corresponding to the syntax of the LWE notifications. The LWE modifies it at each change of document status.

Nevertheless, this solution has some disadvantages. Modifications of handles occur frequently, and each time they require a connection with Global Handle Registry. Besides, the Local Handle Service administration requires additional skills and needs control of other than email transport layer. The Handle System indicates the current location of the document. However, extraction of the list of all the documents in given process requires additional functionality. The Handle System tracks each document separately.

2) Email-based notification system: The ”ground control” service has been also implemented as the email-based notifica-



MIND00001

workflow_process_id	WFP00001
document_id	MIND00001.0001
current_activity_id	ACT00004
sender_of_notification	WRK00002
timestamp	4
deadline	2015-01-15 23:00:00
sent_at	2015-01-03 21:11:17
route-status	received
	history of flow

Figure 8. GUI of the email based ”ground control” service

tion system on basis of email transport layer. Email is intended for frequent passing of messages so that it can easily receive multiple notifications and does not require any additional users skills in the installation, configuration and operation. It does not require unlocking new ports for the transport layer, which affects the security of the organization.

The LWE notifications are sent to one or more email addresses. The notification receiver services run on some organizational server check dedicated mailboxes frequently, parse attached LWE notifications and insert new records to a database.

There are three main tables in the database: Notifications, Documents, and WorkflowProcesses. Each new notification is inserted into the Notifications table. The new notification is distinguished by the address of the mailbox and email's Unique Identifier (UID - a unique number referencing an email in a mailbox). Only those notifications are inserted to the Documents table, which have higher value of logical timestamp than the already registered. The last record for each document ID refers to the current state of this document.

When a notification for a new process appears or process was modified, a new record is inserted to the Workflow-Processes table. This table stores workflow process IDs, the migration path files and workers data files.

Thus, the WorkflowProcesses table stores structure of the process, while the Documents table stores the states of documents flow. The tracing application selects only the most recent records from Documents and WorkflowProcesses tables and constructs a current workflow process structure with its approximate global state.

In contrast to the LWE, which has been implemented in Java, the ”ground control” service has been implemented based on PHP (Personal Home Page) and Postgresql database [20]. PHP technology has been chosen in order to test it for email messaging and XML (Extensible Markup Language) manipulating. PHP provides classes to access mailbox, e.g., by IMAP protocol and functions to XML manipulation. PiBX (XML-Data-Binding framework for PHP) is similar to JAXB (Java Architecture for XML Binding), but it is in the alpha-state at the moment. PHP technology has been good enough for the rapid implementation of the ”ground control” service, but many other technologies could be used for this purpose. In fact, the syntax of notifications is essential for the ”ground control” service, since implementation does not require any new or advanced technology.

Figure 8 shows information about one MIND document selected from the database. An interface allows the user to view all documents related to the process and to view a history of document flow.

During the experiments, emails were received from the dedicated mailbox every minute (for this purpose, the ”Cron” software was used). So emails often appear in mailbox in the different order than they were sent. First sent notification provides a workflow process resources (process definitions, data about workers) to the ”ground control” service. However, sometimes this notification is received later then subsequent notifications. In such a situation, only worflow process ID is inserted to the WorkflowProcesses table and the table is updated at a later time.

Emails with notifications generally were delivered to the inbox without any problems. However, the service does not require to deliver all notifications. However, there was a problem during testing that emails have been received from mailbox and deleted, but the service crashed while writing data to the database. To prevent such situations, emails are stored in the inbox for a month.

VI. Related work

The presented proposal combines existing technologies and new idea to extract some new functionality in the topics of the distributed electronic document and collaborative environ-ments.

The first significant step in the document-based process-ing was the Multivalent Document architecture MVD [21] that introduced active functionality to manipulate a document content with dynamically loaded objects called *behaviors*. The concept of behaviors is similar to the MIND embedded services, howerer MIND expands this concept with local and external services, which can also affect a document behavior, but are not components of the document. This gives documents more flexibility on opening, suiting them better to exploit local resources of visiting devices and to easily add a new functionality.

The Placeless Documents [22] implements document func-tionality with active properties that cannot only manipulate a document content but also manage of a document structure and workflow. The Placeless Documents are reactive, i.e., they respond to external events, while MIND documents are proactive – they initiate their own behavior.

The concept of a proactive document, capable of traveling between computers under its own control has been introduces with a document-agent platform MobiDoc [23]. This platform was, however, closely related to the particular technology, and thus lacked forward compatibility. On the other hand, solutions proposed by MIND found document-agent mobility on stable email messaging standards. Owing to proactive MIND attachments, any email system could be almost like an agent platform with all the benefits of multi-agent systems, but without a need to implement a full-size agent platform that would have to be updated regularly and require additional skills from administrators to run it.

Workflows have been also implemented by WADE (Work-flow and Agents Development Environment) [24] agent plat-form based on JADE (Java Agent DEvelopment framework) [25]. WADE agents embed a micro-workflow engine, capable of executing workflows and compiled before launching the workflow. Performing of activities may be delegated by one agent to another and in principle is not related to agent mobility. This solution follows the classic central workflow enactment philosophy, and differs from it only in decentralization of a global process state into local process states controlled by micro-workflow engines running inside agents. In the MIND architecture, workflow as a XML Process Definition Language (XPDL) file is a part of the whole document and it contains its internal state. LWEs run outside of agents as local workflow engines. Workflow, in the form of plain XPDL, may be also modified during the process execution. Moreover, a document-agent is the only communication interface, making MIND based platforms technologically independent and truly loosely coupled distributed systems.

The reliability of distributed workflows processing is associated with the assurance that the object would not be lost and would arrive to the designated location. It requires some tracking service that in distributed loosely-coupled systems may only estimate the real states of migrating objects. The JADE platform provides some control remote agents (Agent Management System – AMS, Remote Monitoring Agent – RMA) that receives messages from JADE agents, while the "ground control" service has a similar task - it receives messages from the MIND documents to tracking their states. Contrary to JADE control agents, the "ground control" is an external, technologically independent service that communicates with the MIND documents through notifications. Document determines whether the notification has to be sent or not. A syntax of the notifications includes also all document-flow patterns.

VII. Conclusion

Reliable workflow execution of distributed mobile document must be able to handle unforeseen situation when migrating documents fail to reach their destination or get stuck in some worker's device. The "ground control" service, proposed in this paper, is a track and trace service that enables observation of the current document location and stores the history of migration. It allows for checking if the document has reached the recipient LWE or is processed too long on the current device, i.e., if passed the appointed deadline and *sent* notification has not received, it may indicate that the document got stuck in some place. The service does not "tighten" the idea of loosely-coupled distributed system, because it can still execute without this service and the MIND document may decide in which steps the notifications should be sent and in which should not.

Next to reliability, there is also a security issue, which answers the question: what to do if lost document will get to an unauthorized person? The LWE may require authentication of the worker before unpacking and activating document components. The LWE also verifies if the performer assigned to the current activity is the same person as the recipient of the document. The interesting idea has been proposed in [26] by the MENAID (Methods and Tools for Next Generation Document Engineering) project [27]. It introduces a security by the face recognition algorithm built in the MIND documents.

Acknowledgment

This work was supported in part by the National Science Center, Poland, under grant DEC1-2011/01/B/ST6/06500.

References

[1] G. D. Bhatt, "Organizing knowledge in the knowledge development cycle," Journal of Knowledge Management, vol. 4, 2000, pp. 15–26.

[2] L. A. Dabbish and R. E. Kraut, "Email overload at work: an analysis of factors associated with email strain," in Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work, ser. CSCW'06. New York, USA: ACM, 2006, pp. 431–440.

[3] M. Godlewska, "Agent system for managing distributed mobile interactive documents in knowledge-based organizations," in Transactions on Computational Collective Intelligence VI, ser. LNCS 7190, N. T. Nguyen, Ed. Berlin: Springer-Verlag, 2012, pp. 121–145.

[4] M. Godlewska and B. Wiszniewski, "Smart email - almost an agent platform," in Innovations and Advances in Computing, Informatics, Systems Sciences, Networking and Engineering, ser. LNEE, S. Tarek and E. Khaled, Eds. Berlin: Springer-Verlag, 2015, pp. 581–589.

[5] N. Russell, A. Hofstede, W. Aalst, and N. Mulyar, "Workflow control-flow patterns: A revised view," 2006, BPM Center Report BPM-06-22.

[6] Corporation for National Research Initiatives, "Handle.net (version 7.0): Technical manual," 2010.

[7] WfMC. Workflow Management Coalition, "Terminology and glossary," WfMC, Winchester, UK, Tech. Rep. WFMC-TC-1011, Issue 3.0, 1999.

[8] J. Klensin, "Simple Mail Transfer Protocol," RFC 5321, IETF, 2008.

[9] K. Moore, "Simple Mail Transfer Protocol (SMTP) Service Extension for Delivery Status Notifications (DSNs)," RFC 3461, 2003.

[10] C. Lewis, "Overview of Best Email DNS-Based List (DNSBL) Operational Practices," RFC 6471, 2012.

[11] "Java Mail," URL: http://www.oracle.com/ [retrieved: Dec., 2014].

[12] M. Crispin, "Internet message access protocol - version 4rev1," RFC 3501, 2003.

[13] K. Moore, "Recommendations for Automatic Responses to Electronic Mail," RFC 3834, IETF, 2004.

[14] T. Hansen and G. Vaudreuil, "Message Disposition Notification," RFC 3798, IETF, 2004.

[15] "Readnotify," URL: http://www.readnotify.com [retrieved: Dec., 2014].

[16] "Whoreadme," URL: http://www.whoreadme.com [retrieved: Dec., 2014].

[17] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair, Distributed Systems: Concepts and Design, 5th ed. USA: Addison-Wesley Publishing Company, 2011.

[18] J. Spira and C. Burke, "Intel's war on information overload: A case study," 2009.

[19] International DOI Foundation, "DOI Handbook," 2013.

[20] The PostgreSQL Global Development Group, "PostgreSQL 9.4.0 Documentation," 2014.

[21] T. A. Phelps and R. Wilensky, "Multivalent documents: A new model for digital documents," EECS Department, University of California, Berkeley, Tech. Rep. UCB/CSD-98-999, 1998.

[22] P. Dourish, W. K. Edwards, A. LaMarca, J. Lamping, K. Petersen, M. Salisbury, D. B. Terry, and J. Thornton, "Extending document management systems with user-specific active properties," ACM Trans. Inf. Syst., vol. 18, no. 2, 2000, pp. 140–170.

[23] I. Satoh, "Mobile agent-based compound documents," in Proceedings of the 2001 ACM Symposium on Document engineering, ser. DocEng '01. New York, USA: ACM, 2001, pp. 76–84.

[24] Telecom Italia, "Workflows and Agents Development Environment," 2014, URL: http://jade.tilab.com/wade [retrieved: Dec., 2014].

[25] Telecom Italia, "Java Agent Development Framework," 2014, URL: http://jade.tilab.com [retrieved: Dec., 2014].

[26] J. Siciarek, M. Smiatacz, and B. Wiszniewski, "For your eyes only – biometric protection of pdf documents," in EEE'13 - The 2013 International Conference on e-Learning, e-Business, Enterprise Information Systems, and e-Government, Las Vegas, USA, 2013, pp. 212–217.

[27] MeNaID, "http://menaid.org.pl/," 2012-2014, National Science Center, Poland, grant DEC1-2011/01/B/ST6/06500.