TASK Quarterly 28 (1) 2024

Serverless edge computing with mobile documents agents

M. Godlewska, B. Wiszniewski

Gdańsk University of Technology, Faculty of Electronics, Telecommunications and Informatics, Dept. of Intelligent Interactive Systems
11/12 Gabriela Narutowicza Street, 80-233 Gdańsk, Poland

https://doi.org/10.34808/hyfz-f798

Abstract

Exchange of documents as email attachments in collaborative decision making lacks support for keeping track of threads of performed activities, which often leads to email overload and strain. A proactive document email attachment linking the passive content and active services could build reproducibility in collaborative processes, even if workers may tend to exchange emails indiscriminately. The key role in achieving this is played by the migration path embedded in the document code, which when interpreted by the specially designed smart email client can support document steering. Owing to this, the email system can provide a platform for document-centric processing and can introduce self-organization in the dynamically set virtual organization using it. Proactive documents exchanged as standard MIME attachments enable any type of working content with reduced workload and improved work performance, without any substantial investment in the job resources already available in the workplace. Moreover, they enable smooth incorporation of email-based collaboration in a solution that pushes the boundaries of building overall intelligence and enhanced cognition across the cloud-edge continuum.

Keywords:

Document-centric collaboration; Business process automation; Smart objects; FaaS

1. Introduction

In the paper we propose to augment email with proactive document attachments combining passive content with active services and interacting with users, their personal devices and cloud services while performing individual activities in the business process in which they participate. Owing to this, a regular email system could be converted into a kind of flexible multi-agent system dynamically balancing computational resources of a business process implementation within the entire cloud edge continuum [1] and introducing self-organization to email-based document exchange [2].

For decades, electronic mail has been a widely accepted computer mediated communication platform. It enables combining simple textual messages with attachments of any format and structure making it suitable for collaborative work based on document exchange. When doing this in an organized way, collaborators can share, distribute and acquire information to make individual or collective decisions, establish facts or produce new knowledge. Although they may proceed formally with the intent to implement a business process to resolve a specific problem, their organization may be set ad hoc and remain informal and virtual, without the need to engage any specific workflow system. This dynamics is made possible due to the fundamental nature of email, where messages may be sent to anybody, anywhere, anytime and while using operating devices of any type. However, this freedom comes at a cost, which is information overload and strain, often reported by workers when the flow of messages becomes excessive or chaotic. Indeed, email systems lack the means to enforce discipline and structure in the organization.

Below we present in detail our agent-based smart objects, both their internal structure making them proactive documents equipped with functionality thanks to which they can simultaneously be information and interface units [3], as well as mechanisms enabling their migration in the network of mobile edge devices and the implementation of distributed workflow processes of arbitrary complexity [4]. We will argue that our proactive document agents operating at the edge can provide a self-optimizing solution for balancing global cloud resources and local participants' resources, with the latter playing a key role in implementing particularly hard-to-automate parts of the business process [5]. In Section 2, the research problem of the paper is formally stated. It concerns the implementation of business processes based on exchange of documents using an existing email system. Requirements for the proposed solution have been specified based upon the analysis of job demands and job resources in email based virtual organizations.

Messages sent via email are static text files with attachments. This approach is safe and universal. However, the process and functionality definition can also be sent in a serialized and readable form if XML or JSON syntax is used. Section 3 presents the concept of a proactive document agent that can be sent in a static form via email protocols and, after reaching the recipient and receiving appropriate authorization, activate its functionality on the local device.

Section 4 introduces the Mobile Interactive Document (MIND) architecture for such documents attached to email messages and a generic functionality of the smart email client (SEC) which is both a lightweight email client and a container in which document agents can deserialize and carry out their mission. SEC enables workflow to be executed largely based on the information provided by the document. This is possible thanks to the document coordination patterns described in our other paper [6].

In Section 5, a case study of the collaboration process based on MIND documents is presented. It provided us with the basis for several experimental proof-of-concept SEC prototypes, implemented for various desktop and mobile computers and platforms. Section 6 briefly reviews various technology components to contrast them to the proposed MIND solution and discusses the implementability of several process management models using our document-agent platform. Finally, Section 7 concludes the paper and summarizes our future research on expanding MIND document architecture further.

2. Process enactment based on email

A long time after its invention, email is still the most prevalent form of computer-mediated communication by people, who widely accept it as fast, cost effective and accessible, also when exchanging documents within organizations [7]. Unfortunately, with the increasing amount of emails received, workers may find themselves unable to cope with the influx of messages and report information overload and strain, especially when the received emails are redundant or ambiguous. These phenomena have a negative impact on productivity and increase operational costs of the organization [8]. Our motivation to augment email systems to better cope with the above was to find a solution that would not require organizations to replace their existing email systems or business process management practices with something completely new. In that regard, let us first identify job demands and resources in such a setting and next specify what functionality the email-based communication medium should provide to enable straightforward implementation of knowledge processes.

2.1. Job Demands and Resources

A useful tool for assessing job demands in organizations could be the JD-R model [9]. It assumes that working characteristics in the organization can be classified as *job demands* and *job resources*, and that job strain develops if certain job demands are too high when job resources are limited. Given that, the basic job demand for a worker handling emailed documents would be retrieving the delivered message from the mailbox, detaching the document from the message, performing some activity related to its content, and forwarding it next to another worker. However, if the activity is an element of a larger process performed by many workers within the organization, job demands for that worker will be much higher than that [10]. If a single worker is involved in several threads of activities at the same time he/she will be required to keep track of many concurrent activities. It would be difficult to exercise control over these activities when other collaborators are required to complete related activities in other parts of a larger process. Flagging messages as important would not help in this instance, as it is a common practice of email users to mark as urgent any message being sent – with the intention to request that the recipient process it faster. In consequence, if thus marked messages are received in excess, they would probably be disregarded. The worker involved locally in many threads would also be required to keep track of each activity that may extend over time and to control which portion of it has already been completed and what remains to be done. Another requirement would be collation of related items to retain them in one email folder. One difficulty in meeting this demand would be that specific relations between documents may be implicitly determined by the overall structure of the process, so all workers involved in the process must possess the same understanding of its semantics and be able to mark the related messages accordingly.

The research reported in [11] indicated that the demands imposed on collaborators in the email-based settings mentioned above are disproportionally loaded on the recipient, who may often perceive senders as egocentrics – assuming that the recipient will always know what to do with the received message and its attachments, will do it immediately, will always possess the required resources and knowledge and therefore will not have to be specifically instructed or guided. This kind of perceived egocentrism is probably the main source of miscommunication, when the sender is convinced that his/her message is clear, but the recipient does not interpret it in the way the sender intended. Communication of such ambiguous messages would usually require parties to take corrective actions implying additional devoting of time and attention to handle the message again. The advent of mobile devices pushed that disproportion even further, by extending availability and accessibility of recipients to the 24/7 level.

Further in the paper, we consider several coordination patterns for collaborative work based on documents, whose manual implementation by senders using the common functionality of email systems would lead (intentionally or not) to collaborators perceiving them as egocentrics. Typical situations may include:

- The message is sent to a party who is not necessarily the right person for the required job. Clarifying that between parties would involve additional communication and delays in completing the assignment.
- The message is sent in error without any document attached or with the incorrect attachment. Sending then the corrective message would contribute to the increased influx of messages and the recipient's confusion or frustration. If sending the incorrect message goes unnoticed by the sender, the recipient would probably have to revoke or reverse any activities performed thus far relative to the incorrect document.
- The message is flagged as urgent by the sender, without assessing its real urgency; its urgency may vary in time depending on events beyond the sender's control.
- The sender does not know whether the recipient possesses sufficient resources to complete the activity. This may relate to whether the latter has the right tool installed, uses a desktop or mobile device, opens the message during off-time or when out of office, the attached document is password protected, etc.
- The message is ambiguous in terms of what tool would best suit handling the document content. For example, formatting of some text documents would not take into account that the recipient may use an alternative tool to edit it (affecting font substitution, default styles, tabs, page margins, etc.).

Senders may also experience egocentrism of their collaborators, especially when after sending a message they become recipients. Typical situations may include:

- Upon completing the current activity, the sender has to wait for details (to be passed via email, by texting or a telephone call) to whom the document should next be forwarded from another worker involved in a different part of the process. Alternatively, the sender may know to whom the message should be sent but has to wait for "the go" signal from another worker.
- The sender is waiting for the detailed feedback on the document from its recipient; there is no way to enforce

any detailed feedback from the immediate successor or from workers involved in further (more distant) activities in the process.

Resources, which according to the aforementioned JD-R model could buffer the impact of overload and strain and improve implementation of collaborative processes with email, may include the following:

- The knowledge about the process structure is in the system, whereas the knowledge on how to perform the activity is left to the workers. In other words, the system knows to whom each document should be sent and is able to calculate the urgency level of each activity.
- Messages with document attachments are automatically sent by the system, ensuring no forgotten or incorrect attachments are possible.
- The attached document provides some elementary (local) functionality to help the worker to successfully complete the activity.
- Based on the calculated message level of urgency, time of delivery and type of device used to open it, the system can assess whether the recipient should be bothered with the message and possesses sufficient resources to process it.
- When necessary, feedback to each document is implemented as an element of the process.

The above has brought us to the conclusion that the driving force of document based collaborative processes using email could be the documents themselves and that this could be achieved at no extra costs to the workers and their organizations. In particular, the email system should not be replaced by any other system, to allow people to further cultivate their organizational habits of using email as their basic communication medium.

3. Proactive Document Agents

Traditional models of collaborative work based on document exchange assume each document to be a passive object, whose content represents a certain corpus of data in terms of bytes, syntax and structure, to be processed with some dedicated tool. The worker handling the document has to select and activate the tool individually, based on its syntax and the local system resources.

Workers who exchange documents using email do not have at their disposal any practical mechanism to provide explicit or implicit control of the emergent behavior of a collaborative process that they may wish to implement. The question is how to make the given collection of interacting document agents to give rise to reproducible collaboration patterns of behaviors that the individual agents acting alone would not be able to exhibit and to aid workers exchanging email indiscriminately in building self-organization in the process.

If, however, the exchanged documents possess execution capability, they may automatically call the appropriate software function required to work with their content and interact with their human users. Such a document-centric computing paradigm could then involve documents implemented as autonomous agents, which render and use services interchangeably with their human counterparts. If the agents additionally include a workflow component that routes them to the appropriate workers, they may be able to migrate on their own between various execution devices operated by people.

Further in Section 4- we propose to incorporate in the body of proactive document attachments a predefined migration path, which could be interpreted by a specialized "smart" email client capable of enacting a regular process with *activities* and *transitions*. With that, no modifications to the underlying email systems would be necessary, as management of each worker's inbox would only require messages with a specific custom header indicating the proactive attachment to be filtered out and the respective activity to be performed to be selected from the path. Upon completing the activity, the transition would be followed.

As mentioned above, the key functionality of the document agent enabling document-centric collaboration are its ability to execute and to migrate. The former refers to activation of the incoming document agent (object) and calling its services to process the related content brought to the device by the agent, whereas the latter refers to the specific operations to be performed on the document to forward it to collaborators operating other devices.

Services of the activated document agent are called by the local client running on the receiving device and acting as the intermediary between the document and the worker. Three types of services are possible: an *embedded service*, implemented as a piece of code or script that may be executed directly in the local system of the receiving worker's device, a *local service*, implemented as a script that can test for availability and if necessary activate a specific tool installed on the worker's device, and an *external service*, implemented as a script requesting the local system to call a specified remote service.

For security reasons, execution of the aforementioned services is governed by preferences of the actual worker, who may decide whether to allow the document to activate its embedded code or just to open its content with his/her locally installed tools. Of course in either case the attached document is routinely scanned for viruses by the locally available anti-virus tool. A decision to activate a proactive document does not deviate from the typical situation when the recipient of a message has to decide whether to open a plain (passive) document attached to it. Usually, when he/she is sure that the document comes from a trusted source, the decision is to open the attachment[1]. In the solution proposed in the paper, we expanded the range of possible decisions not only based on the personal preferences of the receiving worker, but also on the current location of the device on which the document will be opened, the device performance characteristics, the security parameters of its system software, the quality and security of the available network connection, and several other attributes, whose values define the document *execution context*. Moreover, values of these attributes may be negotiated automatically by the document and its receiving worker's device [12]. A preferences file, prepared by the worker, enables our smart email client to choose the appropriate sets of negotiable attribute values called *modes*: a *private mode* indicates what types of document computations would not be possible (local services/tools are preferred and no embedded services are allowed), a *travel mode* indicates that access to available networks is possible but unstable or not recommended for security reasons (thus external services are not allowed), a *business mode* assuming documents coming from the corporate, trusted network from inside of the worker's organization (both embedded and external services are allowed), and an *airplane mode* with no access to any network (external services are not possible and the received document can further be processed only off-line), and so on.

The mobility of the document agent on the other hand could utilize the whole range of data communication protocols used on the Internet for sending and retrieving email messages, including the lower layer TCP/IP or HTTP/HTTPS data streaming protocols, as well as protocols of higher layers, such as SMTP, POP3 and IMAP.

4. The MIND Model

The proposed document architecture enables autonomous document-agents which can carry both content to be worked on and a description of steps to be performed by individual collaborators during the entire process. The document can dynamically adjust to various execution contexts provided by devices visited during migration. These contexts can change dynamically, depending, among others, on the functionality of a proactive document, the policies implemented by its originator, the operational characteristics of its current execution device, its location and, and the preferences of the relevant worker. Owing to that, our documents can be handled in a number of ways: automatically, by using a code embedded in their body (if permitted by the worker), with the use of local services or tools installed on the relevant device or by external (third party) services if an Internet connection is available or permitted at the time of executing the activity.

4.1. Hub Document

The structure of a hub document, as shown in Figure 1, constitutes in its serialized form a bundle of XML data files containing *content* and *resource* files, which are sent by email in one package. Content files represent a passive document *corpus of data* to be worked on by document services and workers indicated in the migration path. Data may be structured in any way and encoded in any MIME format [13]. Resource files describe a document migration path and its required resources, such as services and workers. They are selected by the process originator, according to the specific format and purpose of the document's content. Upon arrival at its destination, the package is unpacked and resource files are unmarshalled to executable objects. A local working directory structure is created on the receiving device for storing these objects and all unzipped content files.

A service object provides document functionality that makes it proactive. The path component consists of objects of the relevant process – represented as standard elements of XPDL, a workflow definition language used in our experimental implementation of mobile document-agents [14]. They are: activity objects, specifying the respective process activities to be performed and transition objects, specifying the migration path to be followed. Each activity object has one or more performer objects and may refer to worker or service objects alike. The worker object provides the worker's email address, while the service object specifies URIs identifying the related document functionality. In general, each activity may be performed by the worker, service, or a combination of both. The passive document content of the part objects, constitutes the corpus of data to be worked on by the relevant activity object. If multiple incoming or outgoing process threads are involved, the activity object may have a transition-restriction object, incorporating either split or join objects indicating its particular coordination pattern type.

The variety of formats that could be used to represent a document content would make accessing it difficult for the receiving worker. Owing to embedded functionality of a document provided by its service components, such dif-

[1]Contemporary viruses that can infect an electronic document, e.g. PDF, DOCX or ZIP, usually activate some malicious activity on opening its content, making such a document in fact a reactive one.

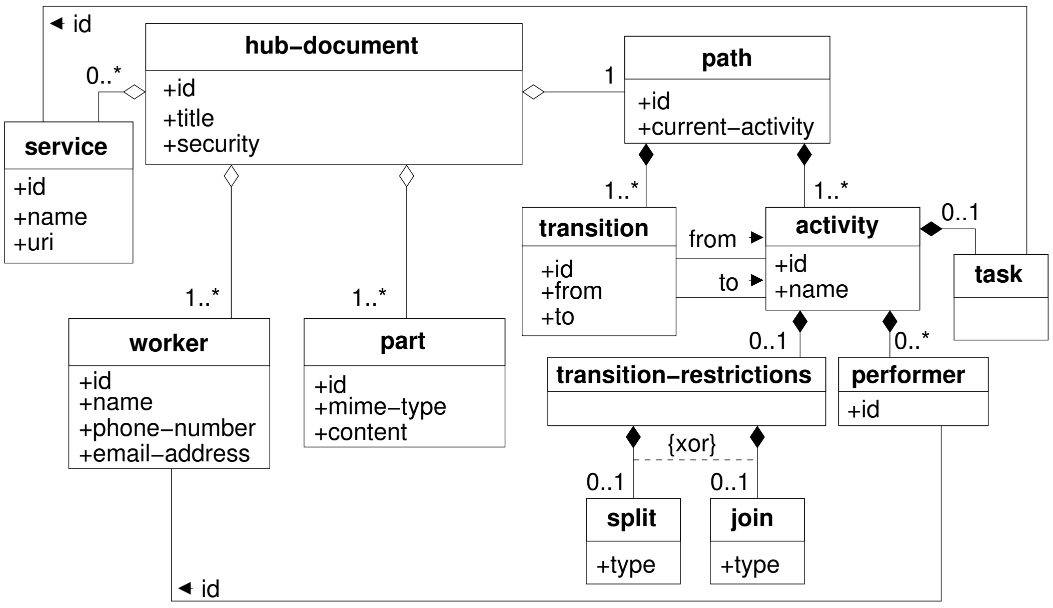


Figure 1: A document-agent architecture

ficulties can be eliminated without making the proposed model dependent on any particular content format. It also simplifies the overall document structure, since the functionality organizing access to document assets (parts) can be coded directly in the document services.

4.2. Smart Email Client

The generic functionality of a smart email client capable of handling proactive document attachments is specified in the diagram shown in Figure 2.

Initially the email message with a packed hub-document is delivered to the respective worker's mailbox on an email server (S1). Each such message is distinguished from other messages stored in the mailbox by a custom header field, followed by a unique document identifier. Depending on the particular collaboration pattern, transition to state S2 may require delivery of one or more messages. A smart email client identifies to that effect all relevant messages in the mailbox and retrieves them to its local inbox. Making document components ready to unpack (S3) may require authentication by the worker. Unpacking of the retrieved messages enables assembling hub-document objects (S4), which entails creation of a local folder structure to hold unzipped content files and deserialized resource objects indicated in Figure 1. Activation of the document enables its embedded functionality, so it may interact with the client, a worker, the local system, and available external services (S5). Interaction begins with obtaining a document path component (S6), determining a current-activity to be performed, and results in resuming a thread of a process on the execution device (S7). Note that the respective activity object includes a task object, which enables interaction of a document with the client and the user via service objects (S8). When the required work on the content of part objects is completed, current-activity is considered complete (S9). States S7, S8 and S9 specify the generic execution states of the activity, distinguished for proper implementation of the milestone and cancel activity patterns specified in the previous section. The client determines the next activity to be performed and the respective worker object responsible for that (S10). The latter specifies the email-address of the worker, indicated by the performer object in the activity object of the path component that follows the completed current-activity. If the worker is the same as before, the content of local folders is refined (cleaned) to prepare document part content objects processed so far for the next activity, which is eventually enabled (S7) for the next cycle. All document components for other workers are serialized accordingly (S11), depending on the particular collaboration pattern specified by the related transition-restictions object (S12) packed in one or more messages, submitted to the respective email server and sent out (S13).

A smart email client can be implemented either as a stand-alone application or a plug-in to the existing email client. In either case it shall be considered a *lightweight* email client, as its responsibility would simply be retrieving and submitting messages with a document-specific header field content, as mentioned before, activating selected objects serialized in the attached document files and making them work in a local system.

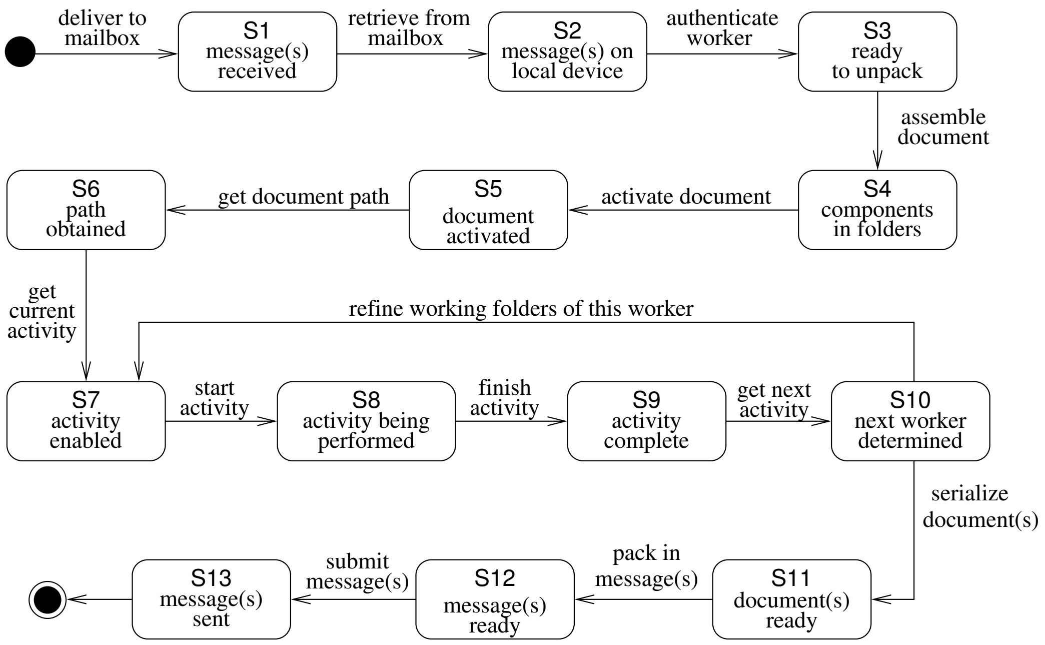


Figure 2: States of a document lifetime on the execution device

5. Experiments

Several proof-of-concept prototypes of smart email clients were implemented by us to handle proactive attachments according to the diagram in Figure 2. They were tested using the class roster application described below. Although it implements relatively uncomplicated decision processes, related to grading student work in a typical academic setting, it is sufficient to demonstrate the non-algorithmic flavor of an interactive computer system.

5.1. Class Roster Case Study

Consider the course grading knowledge process in which instructors judge the academic quality of students' work and assign grades as symbols of their evaluation. In doing so, they collaborate towards assigning a final (semester) grade for each student registered in the course. A Registrar's Office (RO) is the course grade roster document originator, whose collaborator is a Course Leader (CL). CL runs his/her subprocess to successively collect credits from other instructors during all semester weeks. Structure and implementation of that subprocess is irrelevant to RO. While RO may use an on-line grading system for one-time roster submission and approval, CL is free to implement collection of credits on his/her own, in our case with email messages. The dynamics of the grading process involves *scheduled* events, such as assessment of lab assignments and *unscheduled* events, such as occasional grade corrections, which can both be readily implemented with the document coordination patterns defined before.

Figure 3 specifies the course management process from the perspective of RO, to which CL is the only collaborator. Note two *sequencer patterns* – one for sending the roster only with student names and IDs to CL at the beginning of the semester and another for receiving it with final grades at the end of the semester. Details on collecting scores that correspond to assignments listed in the course schedule, in particular their weights and timing, are irrelevant to RO, who perceives grading as a single activity performed by CL. Alternatively, CL can implement that activity using the *internal subprocess* to handle all relevant types of classes selected from the set of lectures, tutorials, seminars, projects or labs, engage instructors running them, and monitor deadlines for each respective assignment. A generic structure of that subprocess is shown in Figure 4.

Sequencer, decomposing splitter, merger, iterator and internal subprocess patterns shown in Figurese 3 and 4 were designed to be executable by SEC. The complete set of such patterns, called document coordination patterns, is presented in detail in our paper [6]. Also the class roster example is described there in more detail to present all the patterns. In this paper we focused on the functionality of SLE as both an email client and a container for document agents.

5.2. Prototypes of Clients

The first smart email client prototype was implemented by us in Java for desktop computers and laptops running Microsoft Windows and Linux. It was used to test the mobility of document agents between physically distributed execution devices. Owing to the universality of common email protocols and the MIME-based standard

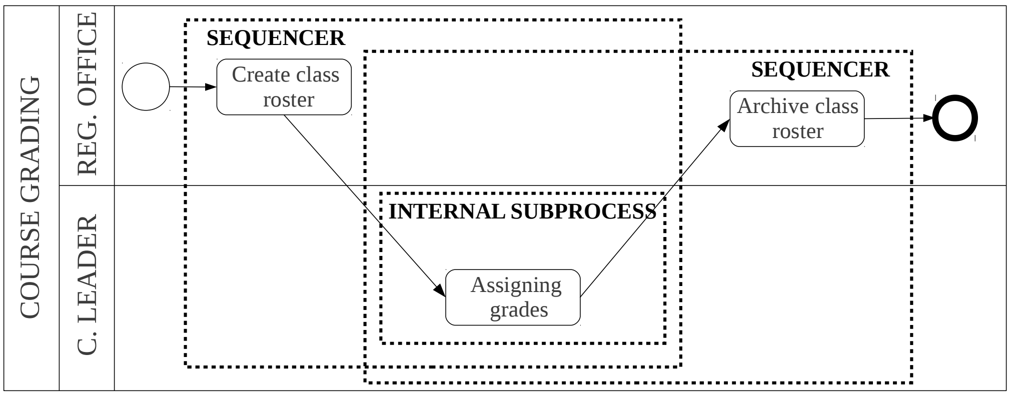


Figure 3: The main grading process

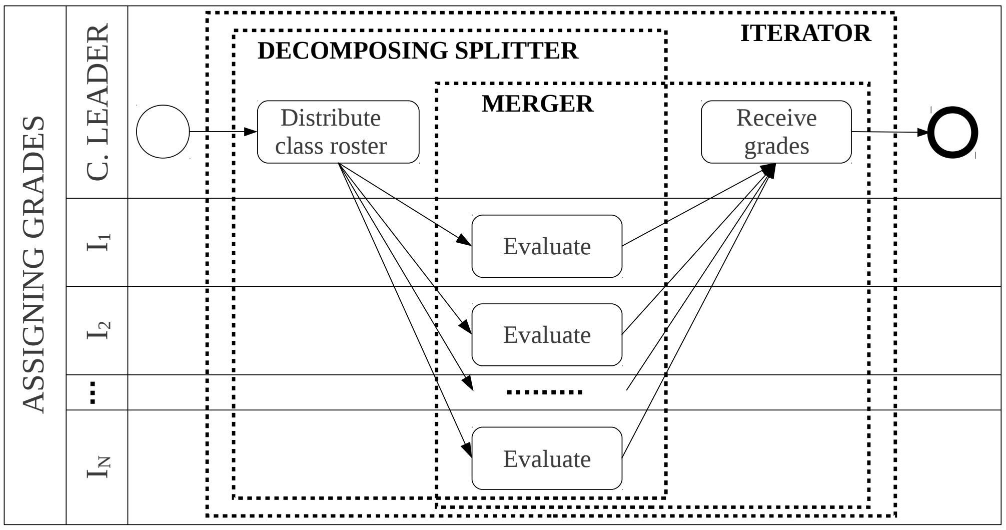
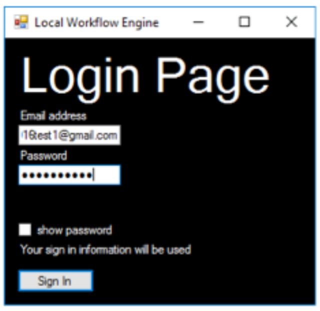


Figure 4: Course leader subprocess

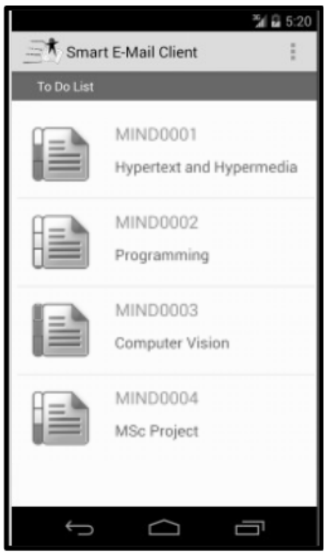
encoding of document attachments [13], our documents can include various data types and freely travel between heterogeneous platforms. Similarly, using XML for assembling hub-document components enabled their activation by binding them to Java objects. That Java prototype was ported next to the Android platform and its textual user interface replaced with the more intuitive graphical touch-up version.

Subsequently, a smart email client prototype was reimplemented for the Apple iOS platforms. The functionality of SLE was preserved regardless of the platform. Besides the operating system, the users may prefer not to activate received documents on their devices. In such a case, the user is free to choose to deal directly with the passive content of the related document. Sample screenshots of our four prototype implementations of smart email clients are presented in Figure 5. They all implement the generic functionality of the client specified by the state diagram in Figure 2, have the same working screen layout, and exhibit a similar behavior.

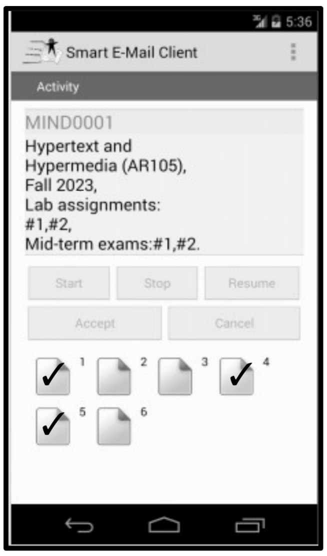
The *Login* page (see Figure 5a) allows the worker to specify a mailbox to log in and access messages with documents to work on. A network connection is needed during that phase in order to retrieve all documents delivered to the given worker from his/her email server. The user can set the option to download only title headers if she/he does not want to download entire documents on a given device or network. The *To Do List* page (see Figure 5b) specifies all threads of which documents have been retrieved to the worker's local inbox. The worker can decide which case to handle first based on the urgency indicated by the color of the case identifiers; they may be respectively 'negligible', 'moderate', 'medium' or 'high'. By selecting the case to work on, its relevant *Activity* page (see Figure 5c) is opened. Depending on the particular collaboration pattern, starting the activity may require receiving more document components still to come. The *Start* button remains inactive as long as any of them have not yet been delivered. In the meantime, the worker may check the state of other pending processes, scroll back to the *To Do List* page and select another case. When all required document components are finally delivered, the *Start* button becomes active. By pressing it, the worker opens the *Content* page and interacts with the relevant document (see Figure 5d). In this particular example, the document is a class roster form requesting the worker to fill in grades



(a) Login page (MS Windows)

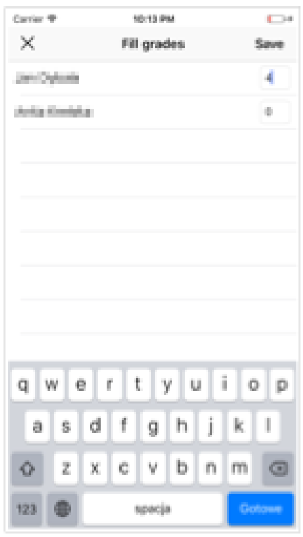


(b) To do list (Android)



MIND0001
Hypertext and
Hypermedia (AR105),
Fall 2023,
Lab assignments:
#1,#2,
Mid-term exams:#1,#2.

(c) Activity (Android)



(d) Content page (iOS)

Figure 5: Class roster application user interface

for all listed students, as explained in Subsection 5.1. Upon completion of the document, it closes automatically and migrates further along its path.

6. Related work

Our concept of a proactive document agent draws on four notions: active documents constituting simultaneously units of information and interaction to their users [3], mobile agents, which are software objects that can migrate from one computer to another autonomously and continue their execution on the destination computer [15], workflow systems, which provide infrastructures to arrange applications automating business processes involving human participants [4], and serverless computing [16].

6.1. Active documents

A Placeless Document architecture [17] implemented document functionality with active properties that could travel with a document, e.g. when sent by email, and provided execution infrastructure capable of performing document-specific activities. It introduced two types of active properties: inline, for intercepting operations attempted by users on a document, and delegates, which could extend document behavior in response to that. In contrast to MIND, a Placeless Document is reactive – it can respond to external events by intercepting operations and delegating interaction to some specialized object associating the document with the workflow. In consequence, the process could not be controlled by the document itself, as the delegate was only a client to some external workflow service.

The concept of a proactive document, capable of traveling from one computer to another under its own control, has been introduced by MobiDoc – a compound document-agent platform [18]. Its document-agents can migrate independently as isolated entities, owing to their embedded code. A dedicated MobileSpaces agent system was designed to support this but soon fell into obsolescence for its lack of forward compatibility. Solutions proposed by MIND attempt to prevent that occurrence by founding document-agent mobility on stable email messaging protocols and services. Owing to the proactive MIND attachments, an email system could bring to the network organization all the benefits of a multi-agent system, enabling implementation of complex collective behaviors with simple individual agent behaviors but without the need to implement a full-size agent platform that would require trained administrators to run it.

6.2. Mobile agent platforms

In the late 1990s, multi-agent systems held great promise of new types of applications to be superior in situations that are non-deterministic or so ill-structured as to appear non-deterministic. After stagnating for a while, the proliferation of the IoT concept over the past decade seems to be reinvigorating research into agent-based computing [19]. Nevertheless, out of many agent platforms that have been developed, just a few survived to date; the survey in [20] proposed five categories of criteria which could be used to assess these platforms as the alternative to email based exchange of proactive documents proposed in this paper and potentially exhibiting the desired characteristics of a job resource for document-centric collaborative work considered in Subsection 2.1 The categories included: platform property, usability, operating ability, pragmatics, and security management, split further for more nuanced characterizations of each agent platform. Platform properties referred to basic characteristics of the platform that the potential developer needs to know to understand its purpose. Usability referred to the suitability of the platform for the construction of agent applications. Operating ability comprised all aspects that should be considered during the agent's execution. Pragmatics indicated whether the platform can provide practical solutions and respectively security management, whether agent-based applications running on it could be secure.

If the virtual organization is to be set up ad hoc from a group of casual collaborators (as proposed earlier in the paper), it would be essential to have a cost-free option to install and configure the required run-time environment. Therefore, consideration should be given to the open source criterion listed in the survey in the platform property category and to the installation and technological maturity criteria listed in the pragmatics category. Document originators and workflow designers would prefer their documents seamlessly interact with performers and external services alike, therefore the simplicity and communication criteria listed in the usability category should be considered too. Another important issue for a proactive document-agent would be its ability to execute on various devices, despite their heterogeneity in terms of operating systems and supported programming languages – the two criteria listed in the survey in the operating ability category. Moreover, if document-centric applications considered in the paper are to be reused in a prolonged period of time, the stability criterion listed in this category should also be considered. In other words, each realistic document agent application should be based on a universal network computing platform, like the Java Virtual Machine (JVM), which can be deployed on possibly every internet enabled device, is executable under any operating system, and does not require frequent modifications as it evolves in time. The platform considered for the implementation of MIND documents should also satisfy two criteria listed in the security management category, namely the end-to-end security criterion, when documents are to be opened for processing and thereafter sent to another worker, and the overall platform security criterion. Finally, the mobility criterion not explicitly distinguished in the cited survey should also be considered, since implementing document-agents as objects that can migrate autonomously with their code and data from one computer to another is essential for the distributed processing model proposed in the paper. In fact most of the platforms analyzed in the cited survey suited better simulation of agents' interaction and their emergent collective behavior rather than their migration.

After taking into account all of the above, only the AGLOBE [21], AgentScape [22] and JADE [23] platforms are worth considering. They all are open source, technologically mature and stable, use Java to implement functionality of their agents, rely on JVM as their agents' principal run-time environment, and are relatively simple to install. However, AGLOBE does not support communication with agents of other platforms, as its applications are designed to form closed systems, dedicated to modeling various real-world scenarios such as solving logistics problems or optimizing cooperation of autonomous robots [21]. Due to this, implementation of the external subprocess could be problematic. The JADE platform enables flexible implementation of agent connectivity using the common HTTP protocol, supports HTTPS for encrypted communication, and allows for user authentication for platform security. It can be used on devices with limited resources, including smartphones, and as an industry-driven product, is the most popular agent platform in the academic and industrial communities.

Our first implemented prototype of MIND used JADE. In order to execute on the given device and to migrate to another one, document agents required a set of special container objects running on all devices they could possibly visit. A device operated by the document originator who started the relevant workflow process required the main container object to start first, to which all other peripheral container objects started on other devices had to register next. In other words, collaborators had to set-up the organization's platform by configuring and connecting execution devices to be used during the entire process before actually starting it. One disadvantage of this was that devices of all involved performers had to be known beforehand in order to properly configure the platform. Moreover, configuring a network of containers to enable unrestricted migration of document agents – when containers were hosted by devices running different operating systems and protected behind various firewalls – was prone to errors and not an easy procedure for ordinary workers. Another problem was related to serialization errors we encountered; when larger documents, with content of a size above a certain threshold in their part components (see Figure 1), were about to migrate to another device (see state S11 in Figure 2). If the serialization error occurred the document agent simply vanished and the workflow process was broken. Handling that exception properly required setting up the platform before starting the related workflow process, as in the case of registering peripheral containers mentioned before and complicated implementation of clients handling MIND documents, compared to their email based counterparts implemented in our prototype.

6.3. Serverless platforms

A provision of the MIND architecture making it possible to call external services from anywhere in the Internet allows document agents to extend their functionality beyond the limitations of personal devices operated by individual knowledge workers, imposed both by their hardware performance characteristics and system configuration. Of particular interest are services for information retrieval and file processing, usually requiring access to large data repositories (e.g. intelligent form filling) or involving computationally intensive analysis of visual or textual content (e.g. automatic translation or sentiment analysis). The diverse specificity of business processes that may be implemented with mobile document agents proposed in the paper leads to rather unstable working conditions for providers of such services implemented in the traditional cloud architecture – where servers constitute a monolithic system containing all business logic. Examples include occasional use of narrowly specialized cloud services, such as formatting a document in accordance with the requirements of a given company, or delegating some activity of one workflow process to another. If provided in such a "monolithic" form, they would not be economically viable for cloud providers due to the need to maintain servers in idle state between processes. A more rational solution, within a more general concept of serverless computing, would be spreading business logic into smaller functions and activating them only when needed. Such an event-driven execution model is known as Function as a Service (FaaS) [16] and provides a perfect setting for implementing specialized applications required to complete business process activities of knowledge workers operating their "lean" mobile devices on the edge with the help of "rich" cloud resources. Developers can simply package their application code in containers for deployment, whereas a cloud provider will dynamically allocate its resources to automatically scale the application [24].

Since the first FaaS platform AWS Lambda was released by Amazon [25], other key commercial competitors as well as open source developers have followed [26]. The widespread availability of these services, together with the possibility of using e-mail as a transport layer for our proactive MIND documents, creates unlimited possibilities for building document exchange systems in knowledge-based organizations.

7. Conclusion

As argued in Section 3, collaborative work based on document exchange by email involves communication of content, meaning effective exchange of portions of information between workers, and coordination of activities performed by them in the business process. In a more general sense, these may be viewed as knowledge transactions, defined as transportation of knowledge objects between two or more communicating workers [27]. This in turn leads to the concept of the knowledge-intensive process, whose conduct and execution depends on various interconnected knowledge-intensive decisions made by process members; they concern the knowledge objects and transactions alike [28]. The MIND solution proposed in the paper enables convenient separation of these two concerns in email-based collaboration: knowledge about the process structure remains in the system, whereas the knowledge on how to perform the particular activity is left to the worker. As illustrated by the running example in Section 5, this can introduce a reasonable level of process management to email based document exchange.

To substantiate this claim, let us refer to the spectrum of knowledge-intensive process classes proposed in [29]. Spontaneous exchange of emails with attached documents and without any coordination of knowledge transactions results in unstructured processes. Although such processes can exhibit a great level of flexibility to collaborators, predictability of the results and repeatability of the order of execution of individual activities could be problematic, as decisions of process participants on this are solely based on their experience and implicit knowledge. In email systems, it would cause overload and strain problems discussed in Section 2. If, however, messages with attached documents could bring any information to guide or instruct collaborators on how to control their knowledge transactions, thus implemented processes would exhibit a loosely structured behavior. Their scope would then be implicitly framed by indicating to the worker the undesired behavior. This is the case of some additional variables carried by MIND documents. The overall process logic would not have to be explicitly defined, but its structured, pre-defined fragments should introduce rules governing at least its essential knowledge transactions. If collaborators need to make ad-hoc changes to the process at runtime, they can add subprocesses – whenever the actual course of action needs to deviate or expand beyond the particular activity. Thus according to [29], the MIND based process would become structured with ad hoc exceptions. Finally, with a migration path built in the document body as proposed in the paper, the relevant process would become structured – with all possible options and decisions that can be made during process enactment captured in a process model defined a priori, which can be repeatedly instantiated in a predictable and controlled manner.

Given the above, it may be argued that MIND email attachments provide sufficient flexibility in modeling the whole spectrum of process classes to the extent limited only by the common sense assumption on avoiding email overload and strain. In particular the structured with ad-hoc exceptions class of processes is interesting, as due to the duality of our proactive document, workers may access/open the document content directly with a local application of their choice or let SEC do that indirectly by invoking document services; they can also modify the document's migration path by adding subprocesses or modifying workers/ performers assigned to specific activities. In that way, both anticipated or unanticipated exceptions can be handled dynamically.

Although the path objects embedded in each MIND document specify the associated process as sequences of activities representing units of work, the document-centric approach proposed in the paper conforms to the data-centric paradigm rather than the activity-centric one. Indeed, according to the criteria defined in [30], a common feature of the former is that the availability of specific data objects (instead of the completion of activities) drives process execution.

Data-driven business process modeling, characterized in [30], involves two notions. One is modeling of behavior to describe how data values are acquired by a data object to perform the action, and the other is modeling of interactions to describe how data objects communicate with one another during the process enactment.

Augmenting email with proactive and mobile document attachments can create a resource-rich and productive work environment for exchanging documents by collaborating workers, considered in the literature a form of successful job crafting [31] – when collaborators try to optimize their current work environment to reduce workload and improve work performance. The major contribution of the paper in this regard has been to demonstrate that buffering of job demands for email based document exchange does not require any substantial investment in the job resources already available in the workplace.

Moreover, the proactive document attachments can move email systems closer to the ideal of ubiquitous computing in the application domain of business process management. Arguments to support this claim can be drawn on five generic principles of ubiquitous systems specified in the literature, e.g. in [32]:

1. *Transparency and openness*; the mobility of the MIND document implied by its migration path and coordination patterns makes all email exchange activities required to implement the related business process transparent to workers. In turn, its proactivity allows process originators to make open implementations, as the MIND document can dynamically adjust to the physical execution context provided by its current receiving device by choosing from all available services. Also, new components can be easily introduced into the process by modifying the document migration path on the fly.
2. *Implicit human computer interaction*; our SECs can free workers from specifying every detail of the interaction required to complete each process activity and make email-based document exchange intuitive and integrated with the overall workplace ecology.
3. *Context awareness*; besides the ability of the MIND document to actively adapt to the physical execution contexts mentioned before, its proactivity also enables it to discover and take advantage of the location, time and other characteristics of the individual worker based on information specified beforehand in the migration path by the process originator as well as gathered by the document itself, e.g. when equipped with the machine learning capability.
4. *Autonomy and self-governance*; the proactive MIND document agent is autonomous, i.e. it can operate independently, free from any central authority and with external dependencies limited to the built-in workflow that can be modified during the process in the self-governance manner implied by the loosely coupled distributed system of SECs and document-agents migrating between them.
5. *Individual and organizational intelligence*; the combination of the embedded, local, and external services of MIND documents can make the latter intelligent in many ways, depending on the required behavior of the document-agent when interacting with the worker during the specific activity, as well as when interacting with other document-agents. Just to name a few tested by us: they can negotiate specific attributes of their preferred execution contexts with their current receiving devices [12], they can verify the biometric identity of their legitimate recipients [33], as well as being able to initiate corrective actions in the system if some undesired events occur [2].

Given the fast evolution of the concept of ubiquitous computing stimulated by recent progress in telecommunications technologies (especially 5G) and the growing need to build a global multi-cloud edge continuum enabling intelligent connection of people, processes, data and things – once called by Cisco "the Internet of Everything" or IoE [34]) – the role of proactive intelligent information objects like the ones we propose in the paper will be invaluable. The more so because their implementation will not require users to change their habits, or developers to create new mechanisms in relation to those already available in the said IoE ecosystem. As argued before, the functionality of any MIND object can be freely extended through its external services implemented with FaaS. Benefits of that will be threefold [16]:

1. *Low latency*; due to the possibility of hosting services on any server in any location, there is no need to use the services of global cloud providers; one or several related knowledge organizations (e.g. local administration bodies) may use their own servers to carry out their business processes, closer to the location of the process participants.
2. *Reduced organization costs*; serverless applications do not remain idle between calls, so their server infrastructure does not need to be active all the time. In the case of the business processes considered in the paper, when individual activities are performed in knowledge organizations by people (e.g. e-Government systems), such server idle costs can be significant [35].
3. *Ease of implementation*; according to the model introduced by Amazon [25] developers just have to upload their applications to a simple storage service (buckets); they may be triggered there by events created by users, such as passing a scheduled time or receiving API requests from user devices or other buckets. Owing to this, either a single function or even an entire chain of functions may be invoked in the nameless "lambda function" manner [36], thus providing all the functionality of the external service designed by the developer – but without the need to design and scale the supporting computing infrastructure.

Currently, we are continuing our work on proactive documents to improve various quality characteristics of their email-based exchange that may be important for specific types of virtual organizations. One example is improving reliability of the system – a "ground control" service proposed in [2], which introduced the tracing capability to the system to cope with soft and hard email message bounces and unexpected user interrupts that may interfere with the regular document flow.

Acknowledgements

The research was supported in part by project "Cloud Artificial Intelligence Service Engineering (CAISE) platform to create universal and smart services for various application areas", No. KPOD.05.10-IW.10-0005/24, as part of the European IPCEI-CIS program, financed by NRRP (National Recovery and Resilience Plan) funds.

References

[1] T. Ogino, S. Kitagami, T. Suganuma, and N. Shiratori, "A multi-agent based flexible IoT edge computing architecture harmonizing its control with cloud computing," *International Journal of Networking and Computing*, vol. 8, pp. 218–239, 07 2018.

[2] M. Godlewska, "Reliable document-centric processing and choreography policy in a loosely coupled email-based system," *International Journal on Advances in Intelligent Systems*, vol. 9, pp. 1–13, June 2016.

[3] R. J. Glushko and T. McGrath, *Document Engineering - Analyzing and Designing Documents for Business Informatics and Web Services.* MIT Press, 2008.

[4] N. Russell, W. M. van der Aalst, and A. H. ter Hofstede, *Workflow Patterns: The Definitive Guide.* Cambridge, MA, USA: MIT Press, 2016.

[5] M. Pawlik, P. Banach, and M. Malawski, "Adaptation of workflow application scheduling algorithm to serverless infrastructure," in *Euro-Par 2019: Parallel Processing Workshops* (U. Schwardmann, C. Boehme, D. B. Heras, V. Cardellini, E. Jeannot, A. Salis, C. Schifanella, R. R. Manumachu, D. Schwamborn, L. Ricci, O. Sangyoon, T. Gruber, L. Antonelli, and S. L. Scott, eds.), (Cham), pp. 345–356, Springer International Publishing, 2020.

[6] M. Godlewska and B. Wiszniewski, "Document coordination patterns," *TASK Quarterly*, 2025.

[7] C. B. Sullivan, "Preferences for electronic mail in organizational communication tasks," *The Journal of Business Communication*, vol. 32, no. 1, pp. 49–64, 1995.

[8] D. D. Dawley and W. P. Anthony, "User perceptions of email at work," *Journal of Business and Technical Communication*, vol. 17, no. 2, pp. 49–64, 2003.

[9] E. Demerouti, A. B. Bakker, F. Nachreiner, and W. B. Schaufeli, "The job demands – resources model of burnout," *Journal of Applied Psychology*, vol. 86, no. 3, pp. 499–512, 2001.

[10] V. Bellotti, N. Ducheneaut, M. Howard, I. Smith, and R. E. Grinter, "Quality versus quantity: E-mail-centric task management and its relation with overload," *Human–Computer Interaction*, vol. 20, no. 1-2, pp. 89–138, 2005.

[11] D. Derks and A. B. Bakker, "The impact of e-mail communication on organizational life," *Cyberpsychology: Journal of Psychosocial Research on Cyberspace*, vol. 1, no. 1, p. article 4, 2010.

[12] J. Kaczorek and B. Wiszniewski, "Bilateral multi-issue negotiation of execution contexts by proactive document agents," *Int. J. Ad Hoc and Ubiquitous Computing*, vol. 32, no. 3, pp. 1–17, 2019.

[13] N. Freed and N. S. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, Internet Engineering Task Force RFC 2045." `http://www.ietf.org/rfc/rfc2045.txt`, 1996.

[14] WfMC, *Process Definition Interface – XML Process Definition Language.* Workflow Management Coalition, version 2.2 ed., August 30 2012.

[15] M. P. Singh and M. N. Huhns, *Service-Oriented Computing: Semantics, Processes, Agents.* Wiley, Jan 2005.

[16] H. B. Hassan, S. A. Barakat, and Q. I. Sarhan, "Survey on serverless computing," *J. Cloud Comput.*, vol. 10, July 2021.

[17] 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, pp. 140–170, Apr 2000.

[18] I. Satoh, "Mobile agent-based compound documents," in *Proc. 2001 ACM Symposium on Document engineering*, DocEng '01, (New York, NY, USA), pp. 76–84, ACM, 2001.

[19] C. Savaglio, G. Fortino, and M. Zhou, “Towards interoperable, cognitive and autonomic IoT systems: An agent-based approach,” in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pp. 58–63, 2016.

[20] K. Kravari and N. Bassiliades, “A survey of agent platforms,” *Journal of Artificial Societies and Social Simulation*, vol. 18, no. 1, p. 11, 2015.

[21] D. Šišlák, M. Rehák, M. Pěchouček, and D. Pavlíček, “Deployment of a-globe multi-agent platform,” in *Proc. 5th International Joint Conf. on Autonomous Agents and Multiagent Systems*, AAMAS ’06, (New York, NY, USA), pp. 1447–1448, ACM, 2006.

[22] M. Oey, S. van Splunter, E. Ogston, M. Warnier, and F. M. T. Brazier, “A framework for developing agent-based distributed applications,” in *2010 IEEE/WIC/ACM International Conf. on Web Intelligence and Intelligent Agent Technology*, vol. 2, pp. 470–474, Aug 2010.

[23] F. L. Bellifemine, G. Caire, and D. Greenwood, *Developing Multi-Agent Systems with JADE.* Wiley, 2007.

[24] X. Niu, D. Kumanov, L.-H. Hung, W. Lloyd, and K. Y. Yeung, “Leveraging serverless computing to improve performance for sequence comparison,” in *Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, BCB ’19, (New York, NY, USA), p. 683–687, Association for Computing Machinery, 2019.

[25] G. Adzic and R. Chatley, “Serverless computing: economic and architectural impact,” in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2017, (New York, NY, USA), p. 884–889, Association for Computing Machinery, 2017.

[26] D. Barcelona-Pons, P. García-López, A. Ruiz, A. Gómez-Gómez, G. París, and M. Sánchez-Artigas, “Faas orchestration of parallel workloads,” in *Proceedings of the 5th International Workshop on Serverless Computing*, WOSC ’19, (New York, NY, USA), p. 25–30, Association for Computing Machinery, 2019.

[27] P. Dalmaris, E. Tsui, B. Hall, and B. Smith, “A framework for the improvement of knowledge-intensive business processes,” *Business Proc. Manag. Journal*, vol. 13, no. 2, pp. 279–305, 2007.

[28] R. Vaculin, R. Hull, T. Heath, C. Cochran, A. Nigam, and P. Sukaviriya, “Declarative business artifact centric modeling of decision and knowledge intensive business processes,” in *2011 IEEE 15th International Enterprise Distributed Object Computing Conference (EDOC 2011)*, pp. 151–160, Aug 2011.

[29] C. D. Ciccio, A. Marrella, and A. Russo, “Knowledge-intensive processes: Characteristics, requirements and analysis of contemporary approaches,” *Journal on Data Semantics*, vol. 4, pp. 29–57, 2014.

[30] S. Steinau, A. Marrella, K. Andrews, F. Leotta, M. Mecella, and M. Reichert, “DALEC: A framework for the systematic evaluation of data-centric approaches to process management software,” *Software & Systems Modeling*, September 2019.

[31] M. Tims, A. B. Bakker, and D. Derks, “The impact of job crafting on job demands, job resources, and well-being.,” *Journal of Occupational Health Psychology*, vol. 18, no. 2, pp. 230–40, 2013.

[32] S. Poslad, *Ubiquitous Computing: Smart Devices, Environments and Interactions.* Wiley Publishing, 2009.

[33] M. Smiatacz and B. Wiszniewski, “Just look at to open it up: A biometric verification facility for password autofill to protect electronic documents,” *Multimedia Tools Appl.*, vol. 80, p. 20089–20124, May 2021.

[34] V. C. Farias da Costa, L. Oliveira, and J. de Souza, “Internet of everything (ioe) taxonomies: A survey and a novel knowledge-based taxonomy,” *Sensors*, vol. 21, no. 2, 2021.

[35] A. Poth, N. Schubert, and A. Riel, *Sustainability Efficiency Challenges of Modern IT Architectures – A Quality Model for Serverless Energy Footprint*, pp. 289–301. 08 2020.

[36] P. M. Kelly, P. D. Coddington, and A. L. Wendelborn, “Lambda calculus as a workflow model,” *Concurr. Comput.: Pract. Exper.*, vol. 21, p. 1999–2017, Nov. 2009.