Towards Integrating Task Control and Shared Artifacts with Google Apps Scripts

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Wirtschaftsinformatik

eingereicht von

Romana Jakob
Matrikelnummer 1227095

an der Fakultät für Informatik
der Technischen Universität Wien

Betreuung: Christoph Mayr-Dorn Ph.D.

Wien, 10. Jänner 2016

Romana Jakob

Christoph Mayr-Dorn

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Towards Integrating Task Control and Shared Artifacts with Google Apps Scripts

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Business Informatics

by

Romana Jakob
Registration Number 1227095

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Christoph Mayr-Dorn Ph.D.

Vienna, 10th January, 2016

Romana Jakob | Christoph Mayr-Dorn

Erklärung zur Verfassung der Arbeit

Romana Jakob
Nabegg 53/2, 3323 Neustadtl an der Donau

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 10. Jänner 2016

Romana Jakob

Danksagung

Danken möchte ich in erster Linie meinem Betreuer, Herrn Dr. Mayr-Dorn, für seine ausgiebige Unterstützung und auch dafür, dass er mich trotz längerer Pause nicht aufgegeben hat.

Vor allem muss ich mich bei dem wichtigsten Menschen in meinem Leben bedanken. Danke Mario, dass du mir immer wieder Mut machst, nicht aufzugeben und mich auch in schlechten Phasen aufbaust und an mich glaubst.

Meiner Familie möchte ich dafür danken, dass sie mich nicht nur während der Schulzeit, sondern auch während des Studiums sowohl finanziell als auch moralisch unterstützt hat.

Acknowledgements

First, I would like to thank my advisor Christoph Mayr-Dorn, Ph.D, for the support and for not giving up on me, despite the long summer break.

My sincere thanks goes to Mario, who is always there for me and trying to encourage me, especially in bad times. Every time I was ready to quit, you did not let me and I am forever grateful.

Last but not least, I want to thank my family for supporting me spiritually not only during my school and studying time but also my life in general.

Kurzfassung

Seit Jahren steigt die Zahl jener, die online ihren Interessen nachgehen. Soziale Netzwerke erfreuen sich steigender Beliebtheit aber auch Plattformen, die zur Sammlung von Wissen entstanden sind, wachsen kontinuierlich. Menschen treten jedoch nicht nur online in Kontakt um Inhalte auszutauschen, sondern auch um gemeinsam Arbeiten zu erfüllen. Die wohl bekannteste und am meisten benutzte Software dafür ist Google Drive, welches Google Docs, Sheets, Slides und Forms beinhaltet.

Arbeiten viele Menschen an einem Prozess, so werden oft Mechanismen zur Regelung der Kooperation benötigt. Eine generelle Lösung, die auf alle Prozesse der Zusammenarbeit anwendbar ist, gibt es jedoch nicht. Meist hat man anwendungsspezifische Lösungen gefunden, die aber nicht in anderen Feldern anwendbar sind. In vielen Fällen greift man dann auf einfache Regeln und Vorgaben zurück, an die sich die Teilnehmer halten müssen.

Diese Arbeit zielt darauf ab eine Architektur zu entwickeln, die über die Dokumente hinaus Möglichkeiten bietet, um die Prozessinteraktion zu regeln. Konkret wird dabei Google Docs mit Skripten erweitert, die mit Services reden, um sich die benötigten Informationen zur Rolleneinteilung und Arbeitsflusssteuerung zu holen. Jeder User hat dann eine Rolle und ihr zugewiesene Berechtigungen, um den Prozess zu steuern. Dies ermöglicht allen Teilnehmern einen guten Überblick, in welchem Status sich der Prozess gerade befindet. Dadurch wird die Zusammenarbeit erleichtert, da jeder Teilnehmer sofort weiß, ob ein Prozess gestartet, terminiert, vervollständigt oder beendet wurde.

Abstract

Over the past decade people and companies vigorously take up conceptional collaboration concepts. Not only speaking about social networking, such as Facebook or Instagram, but also content sharing and joint knowledge creation becomes increasingly popular. Besides the fact that they get in touch online to share contents, they work collaboratively to achieve certain goals as well.

As for processes where various collaborators take part, the interaction usually is guided or controlled in order to achieve collective goals more efficiently. Solutions to this problem are in most of the cases adapted to serve only one specific purpose and are thus not applicable to other fields.

Therefore, this thesis tries to provide a solution with a composable and customizable architecture that can be applied to different frameworks. Specifically, this architecture is applied to Google Docs in order to manage collaborative workflows. This facilitates the working together on artifacts because every participant has the knowledge if the task is already started, completed, terminated or opted out.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Societal changes that have taken place over the past few decades allow new ways of conceptualizing collaboration. Online collaboration is getting increasingly popular, not only concerning social networking but also content sharing and knowledge collection. When it comes to processes where collaborators appear in large numbers, some rules and restrictions or even mechanisms must be applied in order to achieve shared goals more efficiently.

Even companies increasingly utilize collaboration tools to optimize their business processes. Within this field it is even more important to control the workflow to avoid inefficient actions. The use of software supporting collaboration often do not solve all the problems and require some extensions in most of the cases.

So, this thesis tries to provide a solution that simplifies workflow management in artifacts. Therefore, scripts above the artifacts communicate with services to establish process interaction. Concretely, this solution will be applied to Google Docs, to demonstrate an example workflow process.

1.1 Motivating Scenario

Assume a requirement definition procedure in a software development process. The client is interested in collaborating as he/she will later pay for the end result. After a first interview with the customer, the software development team establishes a set of criteria to which the software should be conform to in the end. Neither the client can be completely sure if his/her belongings were understood correctly nor the members of the development team know if it has interpreted the statements of the client in the right way.

One approach to improve the situation is the writing down of the defined requirements. When supported by a traditional document, this process is very probably static and much work lies on both parties when it comes to updating and synchronizing the document as well as communicating inside the requirement definition phase.

Using software for file synchronization would solve this problem partly. Google, for example, offers web-based software office suites within its Google Drive service. Google Docs, Spreadsheets or Slides enable online collaboration in real-time [1]. Managing the requirement definition phase with this software does not sort out all problems. There are still some difficulties, such as the control of the workflow. Neither the client nor the team members know when the process is finished. Of course, this could be communicated over other channels, but this involves on the one hand another software and on the other the process gets distributed, what can complicate the task unnecessarily.

1.2 Problem Statement

As the motivating scenario shall illustrate, collaboration tools enjoy increasing popularity. Bringing along numerous advantages these software solutions also face some limitations. Open questions concerning collaboration tools are listed in the following:

- How can the workflow be managed?
- How do all the collaborators know, when the process is finished?
- How to make the other ones know that the task must not even be started?
- How can different roles be defined?
- How to combine a business process engine with shared artifacts?

A good approach to sort out these difficulties would be to take up existing collaboration software and expand it with new features. Currently, there are some existing solutions, as mentioned in 6.1, but the problem along with them is, that most of them are one-off implementations. Such fixes often serve one specific purpose but are not applicable to other fields.

1.3 Aim of the work

With the motivating scenario and the problem statement in mind, the reader may now see what the aim of this thesis is: Developing an overall architecture applicable to different frameworks, but also applying it to a specific one in order to prove the functionality and outline the possibilities along with it.

At the beginning of the work, literature research was conducted to be up to date with existing solutions. So this thesis also aims to give an review of related work and compare the implementation with existing approaches.

[1] https://www.google.at/intl/de/docs/about/

1.4 Structure of the work

The remainder of this thesis is structured as follows: Chapter 2 provides an overview of related work where the main approaches of collaboration models are discussed. This chapter is concluded by a comparison with the existing approaches.

Subsequently in Chapter 3 the methodology is presented, where an overall architecture is explained apart from the concrete implementation. This is followed by an example use case, motivating the implementation of this thesis. Then, the conceptional architecture is explained, which is described in more detail in the Data Models and Design Methods section.

Afterwards, in Chapter 4 the main work of this thesis, the implementation, is presented. Within this chapter implementation-specific details are discussed.

Then, Chapter 5 evaluates the implementation in form of an example work flow and presents all the opportunities within the framework.

Chapter 6 critical reflects and compares the implementation with related work and discusses open issues.

This thesis is concluded in Chapter 7 with a summary and outlook on future work.

CHAPTER 2

State of the art

Over the last decades a continuously growing number of companies is optimizing their business processes to meet their business goals. Business process models were developed in order to define which activities have to be executed, their pre- and postconditions, the resource usage as well as the duration time. For this purpose a lot of different approaches to model this processes arose, which resulted in numerous process languages.

The definition of a business process has to cover many different aspects such as control and data flow, organizational view etc. A perfect language would serve all these aspects, but unfortunately none of the existing approaches covers all the mentioned fields. To be eligible for different scenarios, modeling paradigms make use of different approaches. Therefore process languages can be divided into categories according their approach. The approach is either task-centric (2.1), artifact-centric (2.2), human-centric (2.3) or the social context (2.4) is of prime importance.

This thesis mainly focuses on the human-interaction aspect and the collaboration of users. In literature, collaboration often appears to be the coordination and synchronization of processes by ignoring human-centric interactions. In the following, different process languages with their advantages and disadvantages with respect to the collaboration of humans in processes are outlined.

2.1 Task-centric approaches

The task-centric approach uses activities and control structures as the main modeling constructs [30]. Activities in this context, constitute pieces of work that form logical steps within processes [39]. It regards data objects in specific data states as pre- and postconditions for activity enablement or as main decision indicators at gateways [30]. Generally, activity-oriented process modeling approaches focus on repeated activities and their structure [35]. They put significant power into the hands of the users for the purpose of flexibly defining and adapting processes. [10]

The main representative and industry standard for the task-centric approach is the Business Process Model and Notation (BPMN) [30].

2.1.1 BPMN

BPMN has a methodical approach to improving an enterprise's business processes through end-to-end implementations [4]. BPMN activities seek to make business processes more effective, efficient, capable and more agile to an changing environment. In classical BPMN, processes are defined centrally by the organization and deployed for execution by internal performers. Actors are formally entitled to execute an activity. [5]

Interactions between actors are represented using the "message flow" concept. Within this concept data is exchanged between two actors of a process. These actors in turn are represented using a "pool" concept. Pools can be divided in many "lanes" which are different roles of an actor. The synchronization mechanisms in BPMN are the following [36]:

- events: start event, intermediate event and end event concepts
- sequencing: sequence flow concept
- forking: parallel gateway concept
- conditioning: data-based gateway and event-based gateway concepts

Many researchers have documented on collaboration in BPMN. In most of the cases, as mentioned before, collaboration is understood in the context of interaction of users among different activities, but not within one activity. Figure 2.1 shows an example for an collaborative B2B Process that shall illustrate the collaboration over lanes but not within an activity. [40]

Although BPMN provides interaction between actors, implicitly only one resource, i.e. the agent, is responsible for the execution of a task. BPMN is so to speak a closed-world approach where one process is only executed by one agent [5]. Collaboration of more users in executing one specific process is not supported explicitly. In its classical way, BPMN defines collaboration not among actors but between an actor and the system. As shown in Figure 2.2 information exchange between an actor and the system is modeled using message flows between the participants. [9]

Therefore, in its conventional meaning, the outcome of a process is produced by one single agent and no collaboration mechanisms are offered by BPMN.

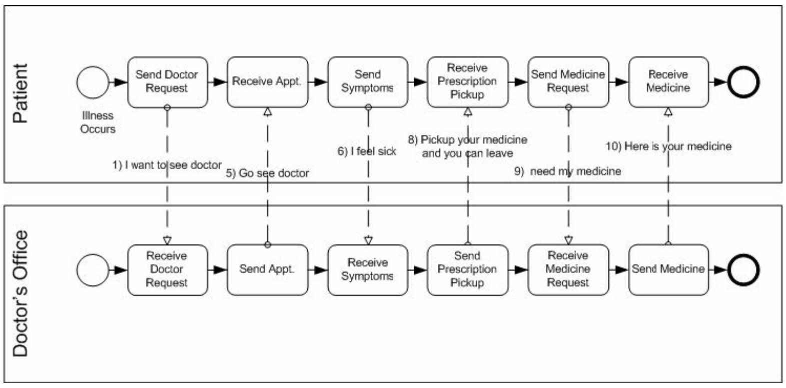


Figure 2.1: Example of a Collaborative B2B Process

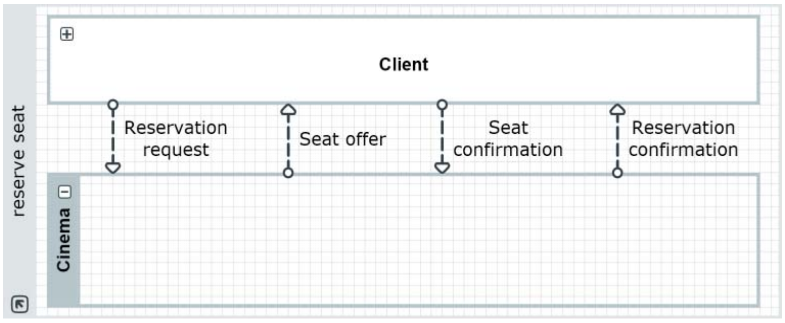


Figure 2.2: Interaction between actor and system as BPMN collaboration

2.1.2 Camunda

Camunda BPM, like its ancestors jBPM and Activiti, is a BPM platform, centered around a runtime engine for executing business processes design using an included modeling tool. It was introduced to close the gap between business analysts' views of business processes and actual software that automates those processes. [22]

This kind of BPM software makes use of some native assignment concepts defined by BPMN. Only one user can be assigned to the task as a human performer. This user is called the *assignee* in the engine terminology. Tasks that have an assignee are not visible in the task lists of other users and can be found in the so-called personal task list of the assignee instead [6]. So the collaboration of more users on a task is not supported.

2.1.3 Specificity frontier [2]

Another good example for a task-centric approach is the **specificity frontier**, introduced by Bernstein. The specificity frontier tries to bridge the gap between automating fixed work processes and the communication in ad-hoc processes. This tool flexibly supports processes at many points along a highly specified to a highly unspecified spectrum. It provides the capability for configuring collaborations depending on a process context.

The development of this tool arose from the problem that an increasingly number of groups and organizations have to adapt their processes due to new customer demands, rapid changes of technology and new competitors. Traditional process-support systems, e.g. ERP, lack flexibility, as they are focused on supporting fixed organizational processes. The only alternative in handling problems that do not fit into the predefined processes is the use of communications support systems, mainly email.

Each active element of a process model is assigned to an agent, which means that only one user can work on a task. The software agent attempts to provide as much support as possible, when it comes to collaborating with other agents in the process model. Agents are going to interact using a speech-act-based protocol, so communication outside of tasks is not supported.

2.1.4 Caramba

Caramba is a process-aware collaboration system. It supports the notion of ad-hoc and semi structured processes as well as combinations and does not require modeling of process templates before enactment. The main focus, however, is on ad hoc processes. Caramba makes it possible for a virtual team to initiate an ad hoc process and to provide a link to a defined process model. The overall idea is to provide process templates in the form of “best practice processes” as part of the collaborative work management environment and to flexibility add to the team members in exceptional cases. [15]

The **Unified Activity Management** (UAM) project at IBM Research [32] has investigated the activity model as a new approach to handle coordination and management in a single (unified) context containing all the necessary resources to execute work. The UAM approach defines the activity structure at a higher level than communication, and thus does not restrict communication. It represents work as an activity pattern which is an initial checklist of activities and sub-activities with an association to their actors and resources. As the activity pattern has no control structure and the checklist in each case is totally under the control of the people carrying out the activity, it must not be mentioned that this approach is an excellent example of a task-centric one.

2.2 Artifact-centric approaches

The artifact-centric approach is often called document-centric, object-centric or data-centric in literature. IBM research initiated the artifact-centric modeling paradigm, followed by several researchers who formalized it further. Additionally, process engines executing such process models have been established and deviations of this paradigm have been developed. [31, 25]

Process-centric business process and workflow models are traditionally unidimensional. The focus lies on the process model, its constructs and its patterns. Support for understanding the life-cycle of the data is left aside, although the data underlies and keeps track of the history of most workflows. In contrast, the artifact-centric approach provides four explicit, inter-related but separable dimensions in the specification of business process: business artifact information model, business artifact macro-level life-cycle, services (tasks) and the association of services to business artifacts. [3]

2.2.1 Business Entity Definition Language [33]

The *Business Entity Definition (BEDL) Language* is a first class representation of data for BPM applications. BEDL can be used alongside commonly accepted process-centric standards such as WS-BPEL and BPMN, in order to bring the advantages of the Business Entity approach into the world of business process modeling. This standard can be integrated into process-centric ones without disrupting the industry investment or the supporting engines and tools. BEDL aggregates access rights, data structure, object state transitions and events. However, the human collaboration aspect remains implicit.

2.2.2 Philharmonicflows [25]

The Philharmonicflows framework implements a well-defined modeling methodology including the definition of processes at different levels of granularity. In order to access data at any point in time, user roles are associated with permissions to create, read, update and delete object instances and their attribute values. In Philharmonicflows different states have different access rights in oder to prevent undesired updates. Shortly, the object behavior is based on states and transitions, as shown in Figure 2.3 [25].

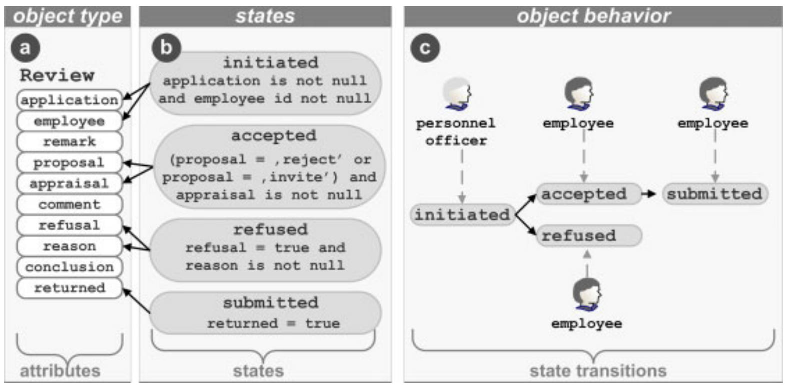


Figure 2.3: Object behavior defined based on states and transitions

For example, after an `employee` from a `functional division` has refused a `review`, he no longer has the rights to change the value of attribute `recommendation`. For this reason, in Philharmonicflows the data authorization and process execution are dependent from each other. So to say, data authorization for a particular object instance needs to consider the progress of its corresponding process instance.

To summarize this section, Philharmonicflows integrates the human aspect in the way of defining one responsible party for one step execution, so the collaboration of humans in one task is not supported.

2.2.3 FlexConnect [34]

FlexConnect is an object-centric process modeling framework and notation designed to support highly flexible processes. FlexConnect focuses on the delivery of human and social services. In contrast to the mainstream artifact-centric approaches, exceptions and variations on a case-by-case basis are the norm of processes. The range of requirements that arise from the ability to support flexible processes are condensed into three patterns of flexibility. Each of these pattern involves a class of users, e.g. social workers or case managers. One flexibility pattern handles the delegation concept which is a moderate approach to handle human-interaction.

As described in [19] task delegation is one specific set of mechanisms ensuring human-centric interactions and supporting collaboration cross organizations. For a better understanding how this delegation concept is carried out in the context of FlexConnect, an example is given in the following.

In Figure 2.4 delegation is demonstrated using a special job object, called *Client Interaction delegator JOB* (job object). Delegation flexibility is achieved by linking a creation region in a JOB to one or more tasks using a delegation signal. The example JOB contains three states: *Make appointment*, *See client* and *Assessment*. The creation region in this example is named *Assessment Region* which contains the *Assessment* state. This creation region imposes two restrictions on the Client Interaction JOB:

1. Delegation from a Client Interaction can only be performed when it is in the *Assessment Region*.

2. Only the subtypes of the Treatment JOB - *Skin Treatment*, *Eye Treatment* and *Mental Health Assessment* - are allowable delegatee tasks from the creation region.

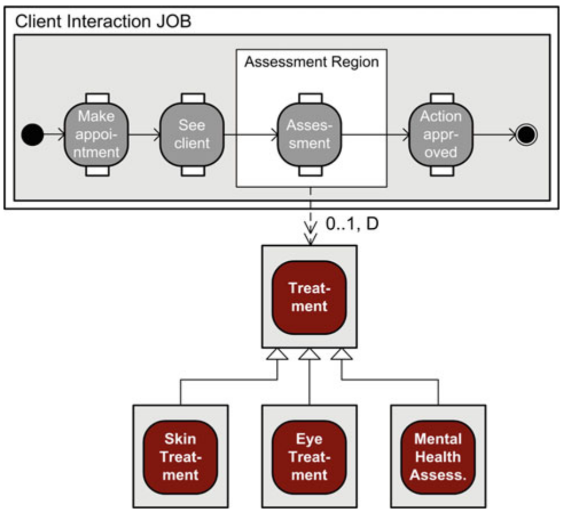


Figure 2.4: Delegation flexibility in FlexConnect

As the multiplicity of the delegation signal is 0..1, delegation is an optional action, so the user decides at runtime whether delegation is performed. It also implies that if multiple instances of a delegatee are needed, multiple JOBs have to be created first, and then each JOB is permitted to delegate as required. It is left to say, that this approach may not be the basic idea of human-interaction.

The traditional FlexConnect framework does not take into consideration which users are authorized to trigger dynamic signals or which users are responsible for acting. This problem can be solved by overlaying an role-based access control (RBAC) model on top of FlexConnect, that allows designers to specify which users, depending on their roles, have the right to trigger signals and which users have the right to resolve referral objects. This approach may not solve the whole human-contribution problem, as it is only limiting the access to steps of the processes.

2.2.4 Ad-hoc processes driven by documents

Complex business processes often rely on intensive information exchange within the company's environment, so they are document-driven by nature [1]. As the title already implies, in document-driven processes documents trigger events in the system. Such events are e.g. document arrival, document updating or document rejection [26]. Agents exchange documents between each other according to the business logic and therefore the document life cycle is managed. However, even when this approach considers documents within the processes, it does not semantically analyses the contents [13].

In [11] the motivating scenario depicts a flexible people-driven order process. The example process is influenced significantly by documents and therefore the process visualization encounters various forms of message flow evolution, such as *missing documents*, *delayed documents*, *premature documents* and *shifted documents*.

Especially in document-driven processes tasks have various kinds of dependencies between them [38]. There are three basic types of dependencies [29]: *Fit*, *Flow*, and *Sharing*. The *flow dependency* describes the state when the output of a task is a required input of another task. A *fit dependency* arise when multiple activities collectively produce a single resource. *Sharing dependencies* arise when several tasks compete for the same resource.

Ad-hoc processes driven by documents lack a solution to the dependency problem, as the collaboration of humans is not stated explicitly. Being an artifact centric-approach it is restricted to them and therefore leaves aside other collaboration mechanisms such as chatting, direct messaging or voting.

2.2.5 The human Architecture Description Language (hADL)

The core of the **human Architecture Description Language** [14] are the collaborators, their means of interaction and dependencies among collaboration objects. The main motivation for hADL is the distinction between following components [12]:

HumanComponents have a particular collaboration role that is essential to the completion of the collaborative effort.

CollaborationConnectors are responsible for the efficient and effective interaction among HumanComponents.

CollaborationObjects abstract from concrete interaction tools and categorize the semantic differences in subtypes.

The separation between those components shall emphasize the difference between the primary collaborating users, e.g. decision makers, and replaceable, non-essential users for coordinating the collaboration, e.g. discussion moderators [14].

However, the other artifact-centric approaches mentioned before have possibilities to model artifacts in much more detail compared to hADL.

2.3 Human-centric approaches

As our environment is guided by technology and the interconnection between people is growing tremendously, there is a call for technical systems, that permit flexible interaction. Humans are following different norms mostly unintentionally, as they are making use of various patterns when interacting in social networks with publish and subscribe mechanisms or using shared artifacts with principal and client roles. Simply put, the human-centric approaches have gained importance over the last years. [10] Although the title already implies, traditional human-centric approaches such as Little-JIL, BPEL4People or WS-Human Task do not support explicit communication among humans outside of tasks.

2.3.1 Little-JIL

Little-JIL is a high-level process programming language with a formal syntax. The main purpose of this visual language is the coordination of agents. A Little-JIL program assists the agents in the completion of a process by assigning steps to the execution agents. Programs in Little-JIL describe the order of steps (units of work) and the communication between them. [7]

Agents in Little-JIL are autonomous entities that are responsible for the initiation of steps and for performing the work associated with them. Agents know how to perform their tasks but can benefit from coordination support. This approach allows the user to clearly express the agent coordination aspects of a wide variety of processes. An agent may be *human* like a programmer in a software development process or *automated*, e.g. a recompilation tool for a ticket reservation system. [20]

By posting a **step** onto the agenda of an agent, the step is assigned to the agent. It is a basic building block of Little-JIL programs and represents a unit of work in the process. One *root step*, representing the entire process is mandatory in every Little-JIL program. A step can be decomposed into sub-steps. Every instantiated step can have one of the five states: *posted*, *started*, *completed*, *retracted* or *terminated*. Optional steps can have a sixth state: *opted-out*. [20]

The implemented approach, as described in the Implementation Chapter 4 of this thesis is based on the described steps of Little-JIL.

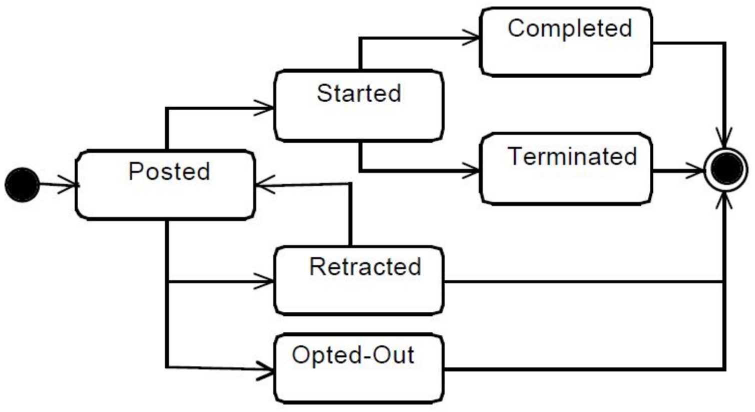


Figure 2.5: States of a step in Little-JIL

In regard to coordination primitives, Little-JIL relies on the process engine as the only coordinator for human involvement. Therefore, this modeling language supports the concept of communication only via passing data explicitly between steps. Despite Little-JIL clearly specifies human work in the process' step definitions, it does not foresee communication among process participants outside of tasks.

2.3.2 BPEL4People [23]

BPEL processes coordinate interactions among different web services. The language encompasses features describe complex control flows, including error handling and compensation behavior. In BPEL4People it is possible to determine who is responsible for acting on a process, a human task or a notification in a certain generic human role. It is stated in [23] that a process definition should incorporate people as another type of participants, because humans may also take part in business processes and can influence the process execution. It is explicitly noted that human interactions are required in many business process scenarios.

This specification introduces the BPEL extension *BPEL4People* to address humans in BPEL as a first-class citizen. It defines a new type of basic activity which uses human tasks as an implementation, and allows specifying tasks local to a process or use tasks defined outside of the process definition. This extension is based on the WS-HumanTask (see next section) specification.

Although BPEL4people lays great importance to the human aspect all interactions are purely task-centric and communication outside of tasks is not supported.

2.3.3 WS-HumanTask

The WS-HumanTask coordination protocol is used to communicate between processes and tasks. With the use of this mechanism state changes are propagated between task and process activities. The process can also perform life cycle operations on the task, e.g. the process can terminate a task [23].

WS-HumanTask defines human tasks, including their behavior, properties and a set of operations. WS-HumanTask uses the WS-Coordination specification as its coordination framework. Human tasks must follow a particular behavior, which requires the definition of a coordination protocol. The WS-HumanTask coordination protocol is used to exchange life-cycle command messages between an application and an invoked human task. [24]

As one can already assume, the coordination protocol is not used for proper communication between tasks. WS-HumanTask only allows attaching comments to tasks, so the main focus is still task-centric. The basic principle of this approach is very close to the one used in this thesis, as stated in the section Methodology. Human tasks can have a number of different states and substates. Figure 2.6 shows a state diagram for human tasks that illustrates the different states and the transitions between them. [24]

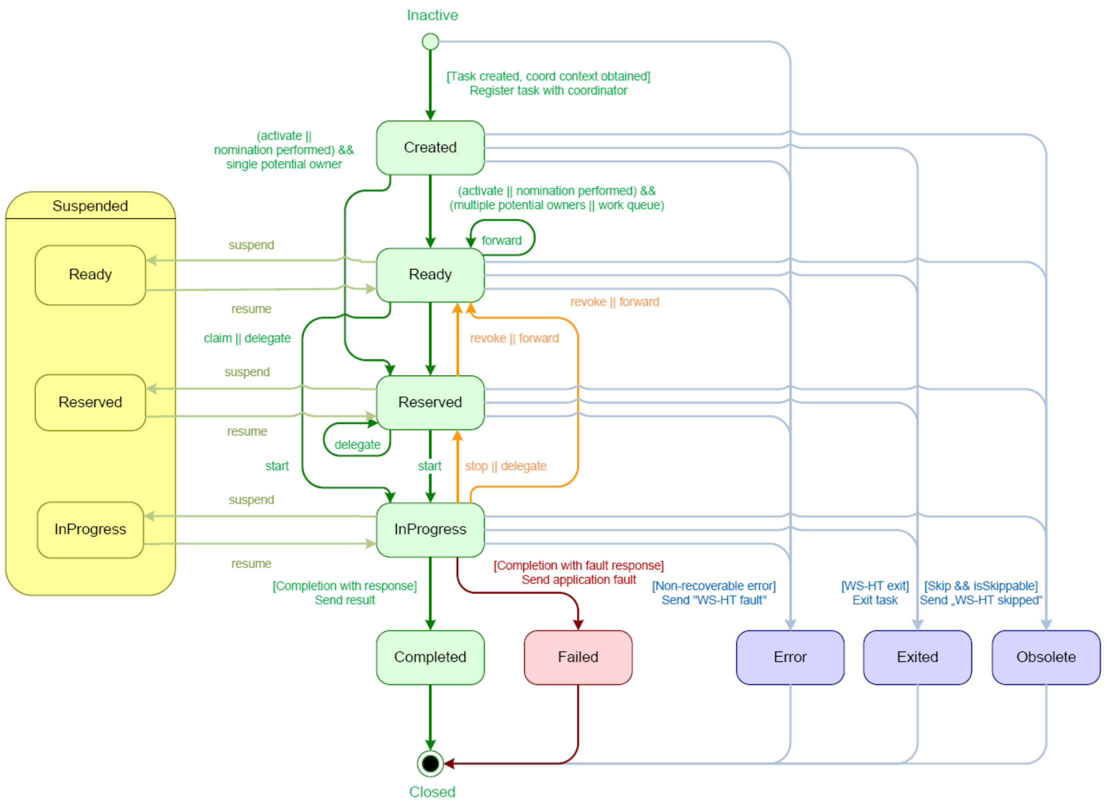


Figure 2.6: State diagram for human tasks in WS-HumanTask

2.3.4 EPC-based Models [27]

La Rosa et al. extend Configurable Event-driven Process Chains (C-EPCs) - a configurable process modeling notation - with notions of roles and objects. The proposed extension supports the representation of a range of variations in the way roles and objects are associated with tasks. A role is dynamically bound to one concrete resource at run-time. A resource can be either human or non-human, but for simplicity, only human resources are considered. The flow of data and physical artifacts is represented through task-object associations. Each object in the process model is statically bound to a concrete artifact, therefore tasks include artifacts.

Despite the involvement of users, the EPC-based model does not specify how multiple users collaborate or how users can interact on artifacts.

2.3.5 Social relations in software artifacts [28]

Contrary to the approaches above, Liptchinsky et al. model the impact of functional or social relations between actors and artifacts. The social aspect is a key principle of this modeling paradigm. Liptchinsky et al. mention that relations between documents, actors and other artifacts may influence the collaboration process. For instance, some tasks should be assigned to actors based on social relations or actions on some documents should not be performed before related documents reach a certain condition. A closer look is taken on non-routine activities, such as the discovery of socially coherent teams or complex decision-making. Therefore, the paradigm promotes modeling not only a network of evolving artifacts, but also an evolving network of people.

However, the collaboration mechanisms that give rise to social relations and process execution support, remain unmentioned.

2.3.6 Subject-oriented BPM [17]

As far as communication is concerned, subject-oriented BPM makes use of data flow diagrams (DFD). DFD represent the flow of data between functions, data repositories and external stakeholders who are not part of the operation of the system. Another diagram used is the Subject Interaction Diagram that illustrates the interaction relationships between the subjects involved in a process. The interaction is conducted via messages between the subjects. These messages contain, if necessary, structured information, also called Business Objects. Therefore the Subject Interaction Diagram is a structured model for subjects with explicit communication relationships. To summarize, subject-oriented BPM models all data flow exclusively with messages between process subjects.

2.4 Social BPM

Social BPM merges BPM with social software, with the aim of providing new possibilities for a more flexible design of business processes and better integration of employees [16].

Performance by means of a controlled participation of external stakeholders is enhanced to process design and enactment. Integrating social software and BPM helps organizations to make use of the value of informal relationships and weak ties. The integration may not have side effects to consolidated business practices embedded in conventional BPM solutions [5].

As mentioned above, classical BPM has a closed-world approach. This can be opened with social features at different levels, according to a spectrum of possibilities, as shown in Figure 2.7. In the following the different levels are shortly outlined, as [18] proposed.

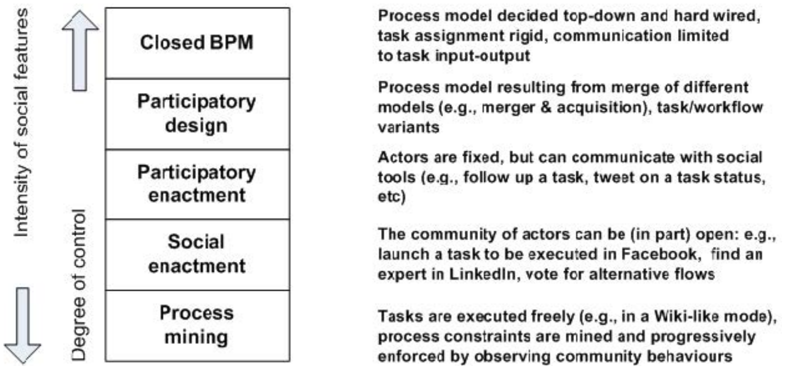

Intensity of social features
Degree of control
Closed BPM
Participatory design
Participatory enactment
Social enactment
Process mining
Process model decided top-down and hard wired, task assignment rigid, communication limited to task input-output
Process model resulting from merge of different models (e.g., merger & acquisition), task/workflow variants
Actors are fixed, but can communicate with social tools (e.g., follow up a task, tweet on a task status, etc)
The community of actors can be (in part) open: e.g., launch a task to be executed in Facebook, find an expert in LinkedIn, vote for alternative flows
Tasks are executed freely (e.g., in a Wiki-like mode), process constraints are mined and progressively enforced by observing community behaviours

Figure 2.7: The continuum of Social Business Process Management

The top of the spectrum, the *Closed BPM* is followed by *Participatory Design* which opens the process design to multiple actors, including end users. Either the stakeholders participate to the definition of the process model or multiple process versions are merged into one shared process model.

Participatory enactment lies the focus of socialization on enactment rather than on design. Despite the fixed actors, as in closed BPM, the communication is not restricted to the input and output of activities. Only actors known at design time, such as internal observers are allowed to interact indirectly with the process, e.g. observation via the intermediation of messages and artifacts.

Social enactment is enabled to support collateral communication, such as following up the status of tasks, commenting on the result of task execution, voting on quality of service, etc. This entails opening the process execution, at least partly, to unknown actors (external observers) at process deployment time and allowing the collective execution of a task.

Eventually, the less structured approach is the *Process Mining.* At this level activities are executed freely and the process constraints are recovered a posteriori, by observing the behavior of the actors such as inspecting execution traces [18].

Brambilla et al. [5] present similar design patterns for the integration of social network features in BPMN as Fraternali's [18] but focuses on participatory and social enactment and extend the existing approach with the main socialization goals, a gallery of design patterns, that represent archetypal solutions to recurrent process socialization problems and a technical framework for generating Social BPM applications.

Dengler et al. [8] criticizes that social software is still insufficiently used as a work resource due to a low integration into workflow management systems. The outlined approach utilizes Wikis and social networks for coordinating collaborative process activities. The information stored in Wikis and social networks is used to find appropriate collaborators from internal as well as from external organizations. As the usage of social software within business process activities requires coordination mechanisms, Wikis are used. But Wiki pages that serve as input for the visualization of process models need to be consistently updated, so the user needs to put in a lot of effort into this approach. Cooperatively performed activities which are supported by social networks need to be supervised and managed. In case of missing coordination support, it is again left to the user to perform the corresponding tasks.

2.4.1 oBPM [21]

Opportunistic Business Process Modeling (oBPM) introduces a new paradigm for modeling and executing business processes that is adequate for process mining, agile process modification and opportunistic task scheduling. Opportunistic BPM stands for designing processes with a minimum of control flow by modeling the states of artifacts involved in business processes.

Furthermore, oBPM includes a top-down as well as a bottom-up perspective that both allow seamless modification to the same model. So this modeling paradigm makes it unnecessary to synchronize multiple models, while still combining the advantage of the two model perspectives.

Contrary to the simple user-centric ad-hoc processes modeled with BPMN, oBPM defines the dependencies between all these user-centric processes without complicating the model unnecessarily. This results in a fine-grained model adequate for agile and simultaneous modification. oBPM also allows opportunistic task scheduling where users can decide what to do first.

However, oBPM is not meant to support unstructured communication and information sharing. oBPM relies on task and artifact abstractions for coordinating business process modeling. As the oBPM model is both user- and artifact-centric, it distinguishes from the approaches mentioned above.

2.5 Analysis of existing approaches

In this section the above mentioned approaches will be analyzed shortly, in terms of human collaboration and the implementation approach of this thesis.

In task-centric approaches (2.1), a lot of power lies in the hands of the users so that they are able to define processes flexibly. BPMN as well as Camunda and the Specificity frontier only allow one user per task. Only Caramba allows collaborative processes as it handles virtual teams.

The artifact-centric approaches (2.2), such as the Business Entity Definition language, Philharmonicflows, FlexConnect and ad-hoc processes driven by documents restrict the access to artifacts, so other collaboration mechanisms remain out of scope. hADL identifies work-centric human components and coordination-centric connectors, so the collaboration aspect is better supported than in the other artifact-centric approaches. Contrary to this, hADL does not have the possibility to model artifacts in such detail compared to the other approaches.

The problem with the human-centric approaches (2.3), such as Little-JIL, BPEL4People, WS-HumanTask, EPC-based Models and Subject-oriented BPM is that all interaction is purely task-centric.

The social BPM approach (2.4) is a typically hard-wired social media connector with no abstraction. The collaboration aspects of oBPM only support the process design phase but as far as its focus is concerned, it is not only task-centric but also user-centric.

CHAPTER 3

Methodology

In this chapter the methodology including the used concepts is described. Firstly, the approach is outlined, followed by the use-case description. Then, the conceptional architecture is explained, which is described in more detail in the Design Methods section.

3.1 Approach

At the beginning of the thesis, there was the problem of how multiple humans can collaborate easily on a task to handle it successfully. Especially, the collaborative work on shared artifacts with easy workflow control and communication methods should be realized with a framework.

To represent the problem pictorially, an example use case was defined which immediately raised some requirements to the solution. The first draft of the example use case was done on paper, afterwards it was modeled with the help of BPMN. At this point, the first problems arose, because BPMN lacks the possibility of illustrating collaboration of more users on an activity, as mentioned in Chapter 2 in the subsection BPMN 2.1.1).

Afterwards a comprehensive literature research was conducted in order to find a modeling approach that covers the revealed requirements in the best way. With the knowledge of the current literature the implementation work began.

3.2 Example Use case

As mentioned above the modeling of the example use case was done with the help of BPMN. Figure 3.1 shows the example use case of a software development process with shared artifacts.

The software development process was chosen as use case, because in this field, shared artifacts could be easily used inside as well as outside the company, when it comes to

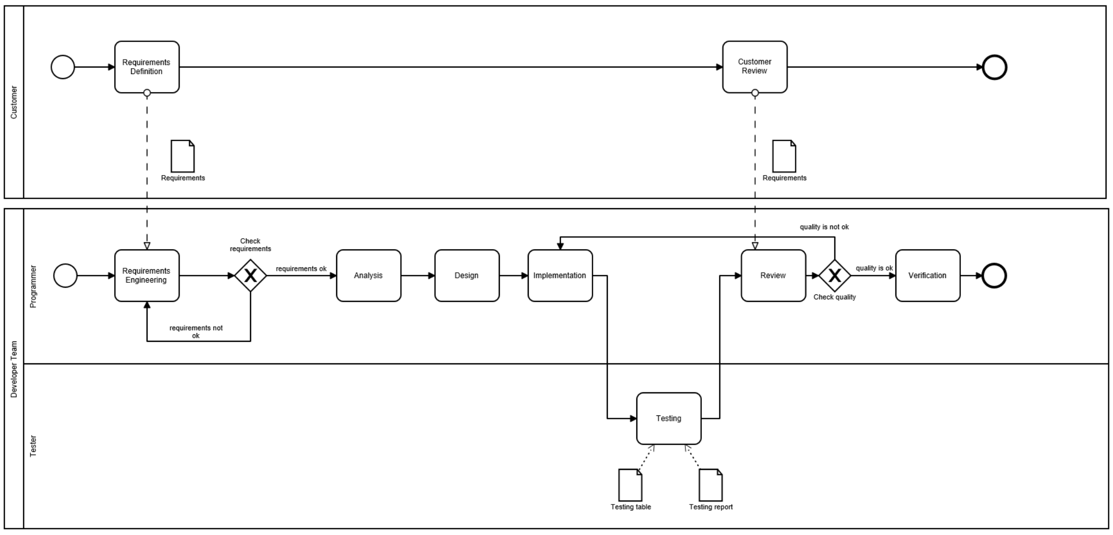


Figure 3.1: Example use case of a software development process with shared artifacts

customer relationships. Figure 3.2 shows the activities that constitute the full life cycle of a software development iteration [37].

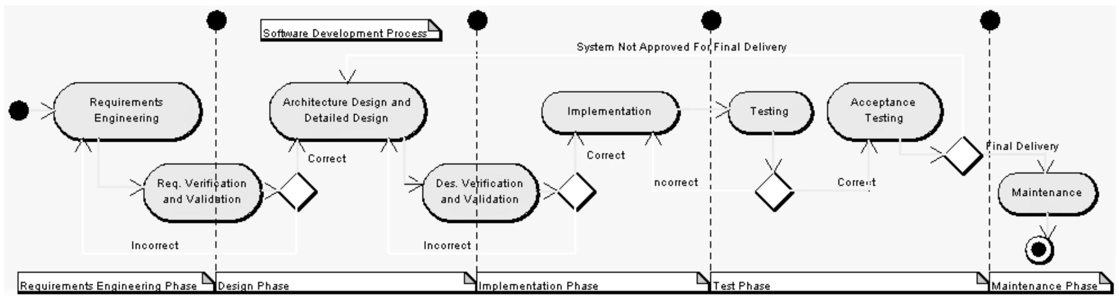

Software Development Process
System Not Approved For Final Delivery
Requirements Engineering
Req. Verification and Validation
Incorrect
Architecture Design and Detailed Design
Correct
Des. Verification and Validation
Incorrect
Implementation
Correct
ncorrect
Testing
Acceptance Testing
Correct
Final Delivery
Maintenance
Requirements Engineering Phase
Design Phase
Implementation Phase
Test Phase
Maintenance Phase

Figure 3.2: Steps of a software development process

In the **Requirements Engineering Phase** shared artifacts are likely used for gathering requirements. This could be done in cooperation with the customer. The result of this step would be, for example, a Google Document to which both parties have contributed and confirmed the conclusion of this phase.

After the check of the requirements is conducted successfully, the **Design Phase** and the **Implementation Phase** are carried out. In these phases simultaneous editing of shared artifacts obviously may not be used.

In the **Test Phase** shared artifacts, for example Google Docs or Google Spreadsheets, can be used when it comes to test reports, test tables or test plans. The example use case contains a review activity in which the requirements are compared to the software product, to verify completeness. Shared artifacts can be used at this step again.

3.3 Conceptional Architecture

The architecture shown in Figure 3.3 is conceptual as it can be applied and used in different frameworks.

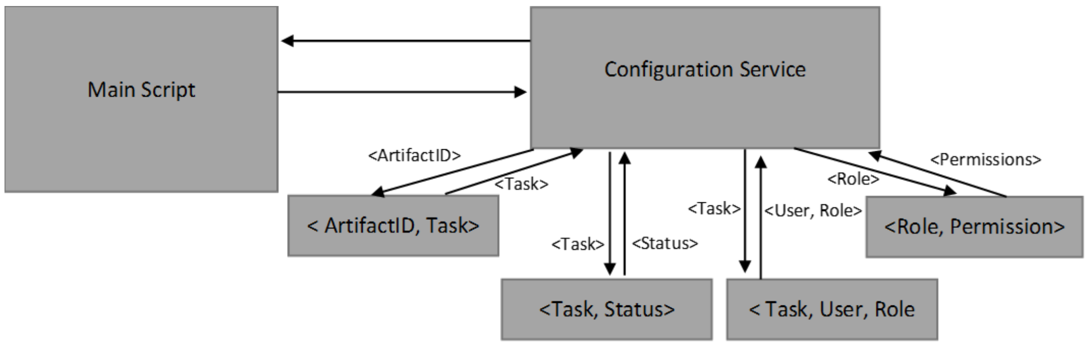


Figure 3.3: Conceptional architecture

It is composed of the following elements: The **Main Script**, the **Configuration Service** and the **Data Structures** containing the information needed for collaboration.

The central collaboration logic is contained by the **Main Script**. This script handles user requests and interacts with other scripts whenever necessary. Every time the artifact is opened by any user with edit rights the main script activates a trigger. The function called by the trigger creates the necessary user interface elements for the collaboration. At this time the user can be informed by the Main script that a script is running in background and how the interaction may work. The UI-elements help the user to interact with the script.

The second step in the collaboration process is the gathering of mandatory information. This step may be triggered automatically or manually by the user, when he or she decides to run the collaboration script. Therefore the Main Script interacts with the Configuration Service to get the necessary information. At first the main script identifies the user interacting with the script, as well as the identification of the artifact. After that the Main Script sends a request to the Configuration Service with the artifact's ID as parameter. As a consequence the Configuration Service addresses an external data structure to get the task associated to the artifact. The task is saved as a global variable as it is needed again later. The same mechanism of request and response is carried out to get status of the task and the user in combination with its role. The task is send as a parameter to the Configuration Service which addresses an external data structure in response to return the status of the task or the user and its role attached to this task. All the returned values are also stored in global variables. The last missing information are the permissions for the user and the task as a unit. The request contains the beforehand role and sends back the permissions for the specified role.

An attentive reader may have noticed that a restriction has been applied to the data structure of artifacts and tasks. The multiplicity of a task to an artifact is one to one, so one document can only be assigned to one task. This design decision was taken due to simplicity. Enabling more documents for a task involves more work for the user, as he or she should notice at first sight to which task the artifact is assigned. Allowing a 1:n multiplicity may also confuse the user, if he or she does not consider that the very same artifact is used for different tasks.

3.3.1 Status of tasks

In order to solve the collaboration problem and to control the life-cycle of an artifact, a task can only be in predefined states. The states are similar to the one that the state machine of Little-JIL supports [20]. For a better understanding, the state machine of Little-JIL is shown again in Figure 3.4.

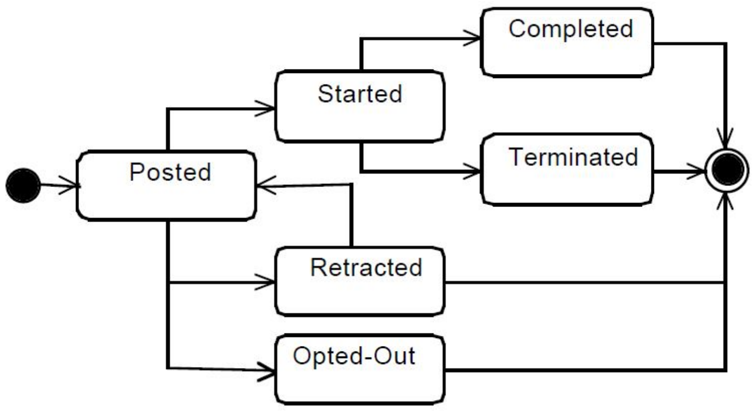


Figure 3.4: State machine of Little-JIL

After instantiation the task is in status *posted*. In the next step any user with the right permissions can decide if he or she wants to start the task or if he or she wants to opt out. The step *retracted* is not supported in the implementation as this step can only be done by the process engine and not by the user. A framework that supports this step must implement a callback mechanism, because the script must therefore listen permanently if the process engine retracted the task.

If the task is *started*, then the user, again with the appropriate rights, can either *complete* or *terminate* the task to which the artifact is assigned to.

After *completing*, *terminating* or *opting out* the task is finished. In our case the task can be reposted again, to allow the user to revise the specified task.

3.4 Data Models

As the data addressed by the Configuration Service is not saved directly into it, the data structure is interchangeable. The data can be stored in CSV, TXT, JSON, XML or whatever desired format, since this architecture makes use of the abstraction mechanism and the functions of the Configuration Service may accept data in whatever format provided.

Google Apps Scripts is a scripting language based on JavaScript, therefore the data format used in this implementation is the JSON format, because of its derivation from JavaScript and thus perfectly interacts with it.

3.5 Design Methods

The presented architecture in this thesis is used within the Google collaboration mechanisms, more precisely Google Docs. As this architecture is quite abstract it may be appropriate for different other frameworks supporting collaboration mechanisms. If a framework provides similar possibilities as Google Apps then it should be easy to implement this approach, according to the demonstrated structure. Frameworks lacking the possibility of collaboration may not be appropriate. To give some examples, an excerpt of possible software is provided in the following:

- Adobe Acrobat
- GoDrive
- Google Apps
- IBM Lotus
- Microsoft SharePoint
- Office 365
- SAP NetWeaver Portal
- Xaitporter

The software do not have to support a collaboration mechanism with text files, also Wikis or tables would profit from the architecture.

CHAPTER 4

Implementation

This section outlines the concrete architecture of the implementation applied to the framework of Google Docs including an explanation of the operations. Afterwards, it is explained how to use the collaboration script.

4.1 Architecture

In order to solve the human collaboration problem with shared artifacts, the architecture shown in Figure 4.1 was developed.

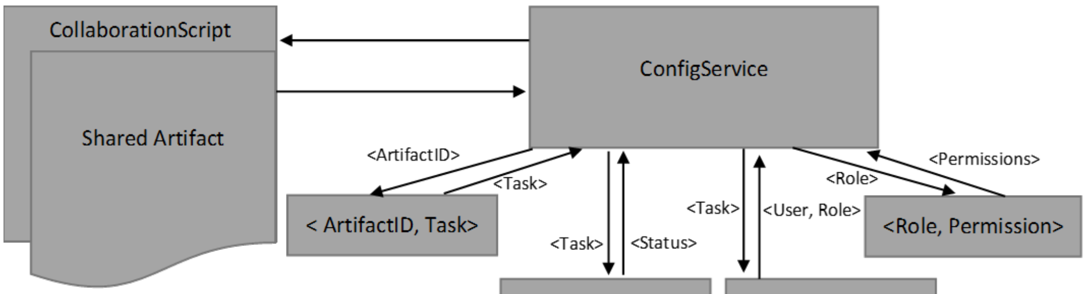


Figure 4.1: Architecture of the scripts plus data structures

Similar to the conceptional architecture presented in the previous chapter, the main components are the **CollaborationScript** the **ConfigService** and the **datastructures** which are in this case contained by the ConfigService.

Referring to the task status explained in 3.3.1, one document is always in one specific state as it is directly assigned to a task. Importantly to mention at this point, is that the solution handles the control flow rather than the data flow.

4.1.1 CollaborationScript

Principally, the logic for collaboration is contained by the CollaborationScript. It interacts with the **ConfigService** to get externally saved data. The CollaborationScript is a bound script.

Bound scripts generally behave like standalone scripts except that they do not appear in Google Drive. They cannot be detached from the file they are bound to and they gain a few special privileges over the parent file. Only users who have permission to edit a document, spreadsheet or form can run its bound script. Collaborators who have only view access cannot open the script editor. If they make a copy of the parent file, they become the owner of the copy, so they are able to see and run a copy of the script.

The **CollaborationScript** contains the following global variables:

var ui saves the user interface for the DocumentApp. This variable is needed to add menu items.

var documentID saves the current ID of the document so that the document can be identified

var urlConfigService saves the URL of the ConfigService [1]

var options saves the options to address data from the ConfigService

The **CollaborationScript** supports the following operations:

onOpen is a simple predefined trigger by Google Apps Scripts that is run automatically when a user opens the document that he or she has permission to edit. This function adds the item "Collaboration" to the menu.

showSidebar creates the user interface element for the sidebar and attaches it to the user interface.

getData calls the functions `getTask`, `getUserRole`, `getTaskStatus` and `getPermissions`, assembles the return values, safes them into a JSON format and returns them to the sidebar HTML file.

getTask creates the URL containing the documentID as parameter, sends an HTTP GET request to the ConfigService and returns a string representation of the task.

getUserRole creates the URL to containing the user and task as parameters, sends an HTTP GET request to the ConfigService, fetches the response and returns the role of the user as a string representation.

getTaskStatus creates the URL containing the task as parameter, sends an HTTP GET request to the ConfigService, fetches the response and returns the status of the task as a string representation.

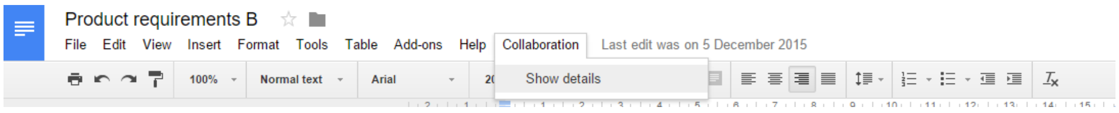
getPermission creates the URL containing the role of the user as parameter, sends

[1] https://script.google.com/macros/s/AKfycbxCaAOkuoOxDa3A1mXv73qJuvOGu4S8RJ-LrXM9DwNf0pkanVIZ/exec

an HTTP GET request to the ConfigService, fetches the response and returns the permissions as an array.

addMenuItem takes the permissions and the status of the task and adds menu items according to this parameters.

Usually, the simple `onOpen` trigger is used to add custom menu items. For security reasons no requests to external services, such as web apps, are allowed in this trigger. Therefore, in the implementation the user must start the collaboration process by selecting the menu item "Collaboration/Show details" (see Figure 4.2) to start the process.

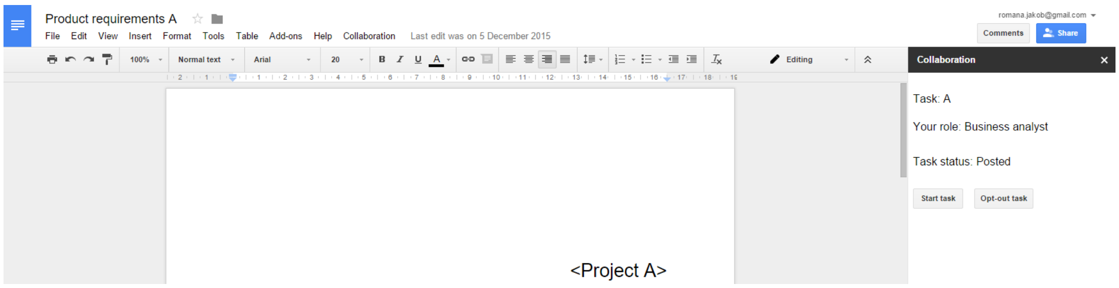


Product requirements B
File Edit View Insert Format Tools Table Add-ons Help Collaboration
Last edit was on 5 December 2015
Show details
100%
Normal text
Arial

Figure 4.2: Menu item to open the collaboration sidebar

Sidebar HTML

By clicking on the "Show details" menu item the `showSidebar()` method of the CollaborationScript is triggered. This creates a new sidebar with the help of the sidebar.html file of the CollaborationScript, as shown in Figure 4.3.



Task: A

Your role: Business analyst

Task status: Posted

Figure 4.3: Collaboration sidebar

The HTML file has some client-side script functions, for example:

```
window.addEventListener('load', function() {
    google.script.run.withSuccessHandler(setData)
        .withFailureHandler(showError).getData();
}
```

This function is an `onLoad EventListener` which is triggered when the HTML-file is loaded. The script calls the server function `getData` and returns the value to the client as a parameter passed to the success handler or in case of exceptions to the failure handler. In other words the handlers are client-side callback functions to run when the server responds. The sideBar contains a classical container with some fields that are filled with the returned values from the CollaborationScript.

As far as user interface elements are concerned, Google Apps Script provide the HTML Service which is an easy way to integrate UI elements. The HTML service can display a dialog or sidebar in Google Docs, Sheets, or Forms if the script is container-bound to the file. Container-bound in this case means, that the script is collocated directly to the shared artifact, like in the presented implementation. The HTML service also provides pre-built alerts and prompts. The way of extending the menu with own items is also used in the implementation, as Google Apps Script offers a lot of Code Snippets that work perfectly.

Other frameworks may not provide such an easy way of integrating HTML-Elements into the user interface. However, the structure in this thesis also works without a sidebar, as far as it contains at least the possibility to integrate menu items or buttons into the predetermined user interface.

Sidebars in Google Apps

A sidebar in Google Apps Script can display an HTML service or UI service user interface inside a Google Docs editor. Sidebars do not suspend the server-side script while the dialog is open. The client-side component can make asynchronous calls to the server-side script using either the `google.script` API for HTML-service interfaces, as shown in the code sample before or server handlers for UI-service interfaces. The sidebar cannot be closed by other interfaces, only by the user or itself[2].

Dialogs in Google Apps

An alert is a pre-built dialog box that opens inside a Google Docs editor. The alert dialog displays a title, a message and per default an "OK" button. Of course, these elements can be changed as desired, even alternative buttons are possible. Alerts suspend the server-side script while the dialog is open. The script will resume after the dialog is closed by the user[3].

4.1.2 ConfigService

The **ConfigService** is deployed as a web app, in order to return data when a request is sent via URL. For instance, parameters such as the status and task can be passed to a URL as shown below:

```
https://script.google.com/.../exec?status=terminate&task=A
```

Web app in Google Apps

In Google Apps Scripts [4] a script can be published as a web app if it contains a `doGet(e)` function that returns an `HTML service HtmlOutput` object or a `Content service`

[2]https://developers.google.com/apps-script/guides/dialogs
[3]https://developers.google.com/apps-script/guides/dialogs
[4]https://developers.google.com/apps-script/guides/web

`TextOutput` object.

The web app **ConfigService** contains following functions:

doGet(e) is called when the CollaborationScript sends an HTTP GET request. The *e* argument represents an event parameter that contains information about the URL parameters. So, the *e* parameter contains the information which query needs to be executed. This function must be contained by the web app in order to be deployed as such. If the parameter *e* contains unknown fields, a TextOutput with the content "Something went wrong." is returned. The query that asks for the status is a special one, because the status is saved into the document properties and not into a JSON data structure, as it must be changeable, contrary to the other fields. Depending on the query, this function calls the other functions to access the needed data.

getRole is called by the `doGet` method when the role of a user is requested. The user and the task are passed as parameters.

getTaskForDocument is called by the `doGet` method when the task for a document is requested. The documentID must be passed as parameter.

getPermissionForRole is called by the `doGet` method when the permissions for a role are requested. The role must be passed as parameter.

getUserRoleTaskData accesses the JSON data structure that contains the mapping between user, role and task.

getTasks accesses the JSON data structure that contains the mapping between documentID and task.

getPermissions accesses the JSON data structure that contains the permissions for a role.

Typically, there are different roles in a software development process. Respectively, two possible roles for a user are defined: The business analyst and the assistant. The permissions for the different states are shown in the Table 4.1.

Status	Business Analyst	Assistant
Start	$\surd$	
Complete	$\surd$	$\surd$
Terminate	$\surd$	$\surd$
Opt-out	$\surd$	$\surd$
Repost	$\surd$	

Table 4.1: Table of permissions for Business Analyst and Assistant

4.2 How to use the collaboration script

In order to use the specified implementation, the CollaborationScript has to be bound to the desired document. Therefore following steps are necessary:

1. Create a new Google Docs or take an existing one
2. Select **Tools/Script editor** in the menu bar to open the script editor. Because bound scripts do not appear in Google Drive, this is the only way to open the script. So in the future, do the same thing, to access the script.
3. Copy and paste the CollaborationScript into the created Code.gs
4. Create a new file called "Sidebar.html" with **File/New/Html file** and copy the code from the provided html file into it
5. Save and close the script editor and reload the Google Document

CHAPTER 5

Evaluation

This chapter shows and example workflow to evaluate the implemented collaboration mechanism introduced in Chapter 4 and to show the possibilities that the different roles have in the various steps.

5.1 Example workflow

The preconditions to take part in the workflow are:

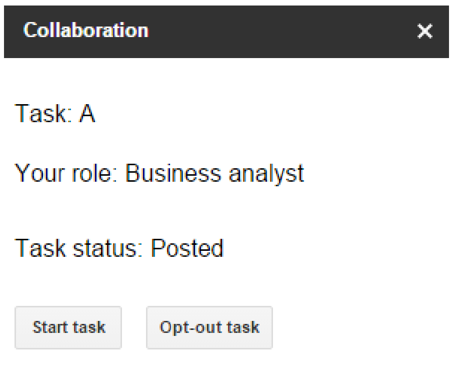
- The document that is planned to use in the workflow must have the collaboration script bound to it, as explained in the previous section 4.2
- The user opening the document must have edit access

When a Google Docs that has the script bound to it is opened for the first time, the collaboration process must be started with the menu item "Collaboration/Show details" (see Figure 4.2). After that the sidebar opens. Depending on the role of the user the sidebar either looks similar to Figure 5.1a if the user has the role *business analyst* or similar to Figure 5.1b if the user is an *assistant*.

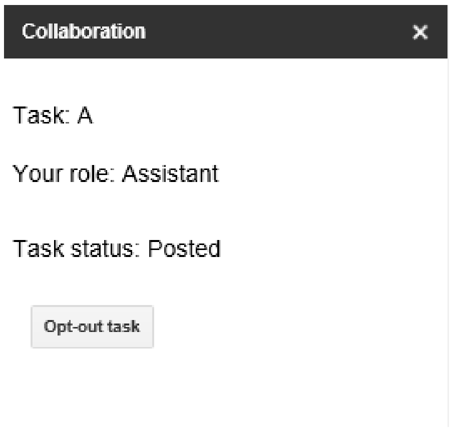
The comparison of the two Figures shows the different permissions that the roles have. As illustrated in Table 4.1. An assistant do not have the possibility to start the task, he or she must wait until the business analyst starts the task.

When the *business analyst* decides to start the task, he or she is asked again if he or she is really sure, see Figure 5.2.

Immediately after the task is started, the *business analyst* gets a notification, shown in Figure 5.3



(a) Sidebar for business analyst



(b) Sidebar for assistant

Figure 5.1: Sidebars when task is in status "Posted"

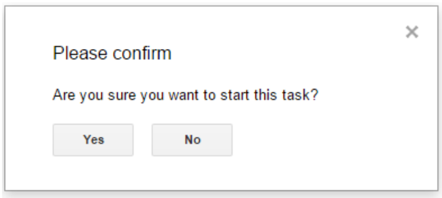


Figure 5.2: Alert dialog to start the task

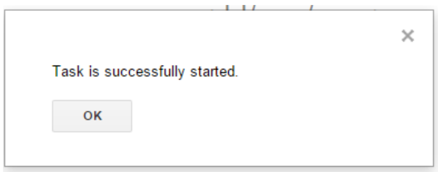


Figure 5.3: Notification of a successfully started task

If the *assistant* decides in the meantime to opt out and has not refreshed his/her page since the business analyst has started the task, he or she gets a notification that the task has already been started, see Figure 5.4. After clicking `OK` the sidebar is refreshed and contains now the updated status and buttons, as shown in the screenshot 5.5b. At this step both roles have the same permissions (see Figure 5.5), so they have the same buttons to go on with the workflow.

Now everyone can either *complete* or *terminate* the task. Again, if one of the both roles have taken the task into another state while the other one has not refreshed his/her page and also wants to go on with the workflow, the slower one gets a notification that his/her action is not possible anymore. Then, the notification is similar to the one shown in Figure 5.3, except the message is adapted to the current status of the task.

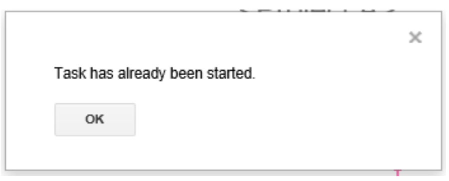
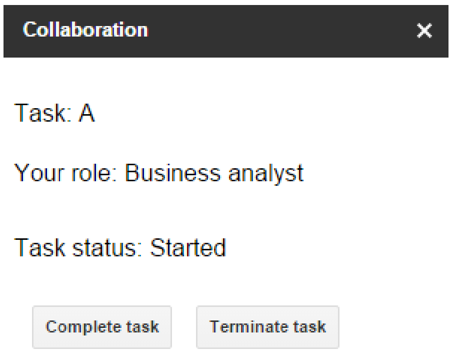

Task has already been started.

Figure 5.4: Notification of an already started task

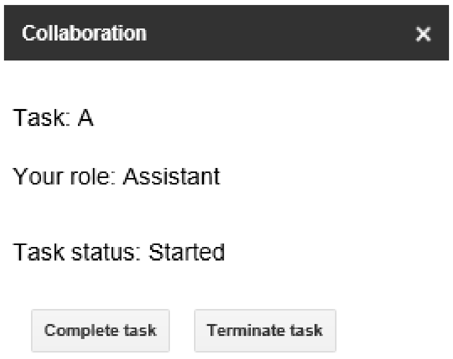

Collaboration

Task: A

Your role: Business analyst

Task status: Started

(a) Sidebar for business analyst


Collaboration

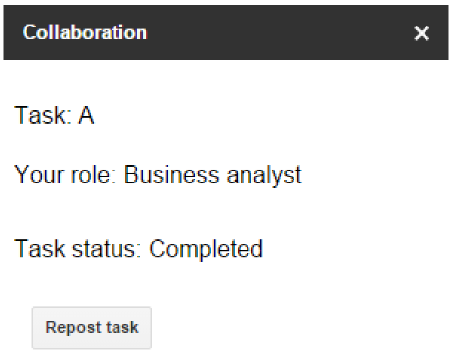
Task: A

Your role: Assistant

Task status: Started

(b) Sidebar for assistant

Figure 5.5: Sidebars when task is in status "Started"

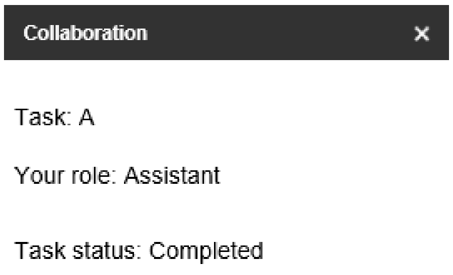

Collaboration

Task: A

Your role: Business analyst

Task status: Completed

(a) Sidebar for business analyst


Collaboration

Task: A

Your role: Assistant

Task status: Completed

(b) Sidebar for assistant

Figure 5.6: Sidebars when task is in status "Completed"

In our case, the assistant decides then to complete the task. After the business analyst has refreshed his/her page, the sidebars look like in Figure 5.6. The business analyst has now the possibility to repost the task, which means, that the task starts again from the beginning.

It is left to mention that the user always has the opinion to interact with the script not only via the buttons in the sidebar, but also via the menu items, as the screenshot in Figure 5.7 shows.

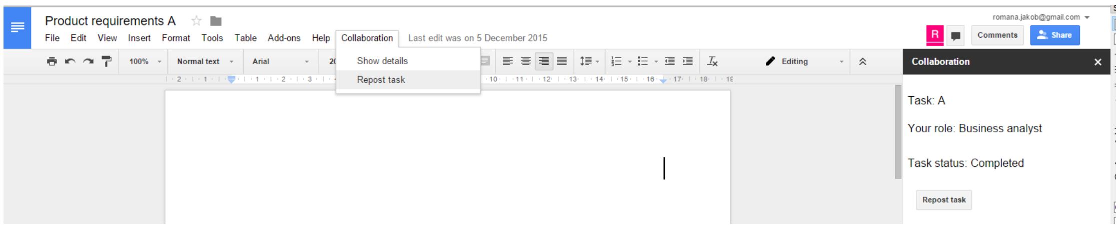


Figure 5.7: Interact with the CollaborationScript via menu items

CHAPTER 6

Critical reflection

This chapter compares the shown implementation with related work and discusses open issues.

6.1 Comparison with related work

The research after existing work in the field of Google Docs, extending collaboration mechanisms was limited almost exclusively to the Google Add-On Store, because similar extensions can be hardly found otherwise.

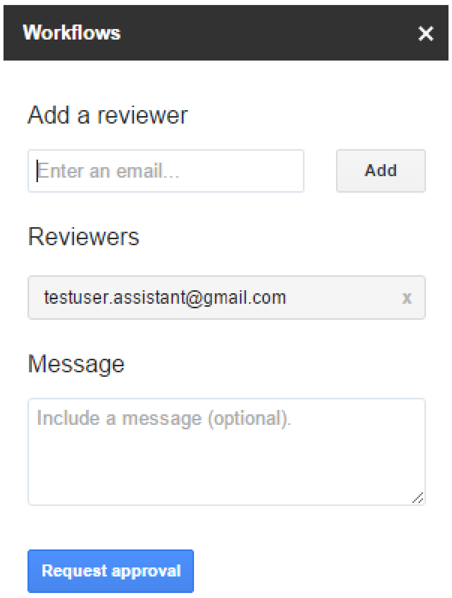
One extension that also handles workflow automation is the Letter Feed Workflow Add-On [1]. This tool helps automating document workflows, as it gathers approvals and feedback in order to deliver it to the owner. A reviewer can be added easily, as shown in Figure 6.1a. This reviewer will be notified per e-Mail and has the possibility to approve or reject (see Figure 6.1b). The owner keeps a good overview of the approval status, illustrated in the Screenshot 6.2a. When the reviewer has approved, the owner gets an E-Mail and the approval status is fulfilled (see Figure 6.2b).

The advantage of this approach is the good overview of the approval status. The presented implementation of this thesis does not give such a good overview of the state. Moreover, reviewers can be easily added in the sidebar whereas the users in the thesis' implementation come from outside.

However, the Workflow Feed example is an one-off implementation as it hardly can be extended with other features. Further, it does not support states for a task such as start, opt out or terminate.

Compared to the existing collaboration approaches shown in Chapter State of the art 2, the implementation of this thesis has its assets. It handles the control flow rather than

[1] http://www.letterfeed.com/workflows


Workflows

Add a reviewer

Enter an email...

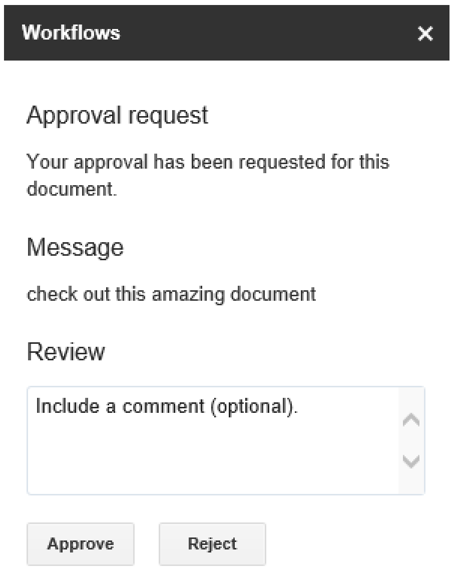
Reviewers

testuser.assistant@gmail.com

Message

Include a message (optional).

(a) Add reviewers for owner


Workflows

Approval request

Your approval has been requested for this document.

Message

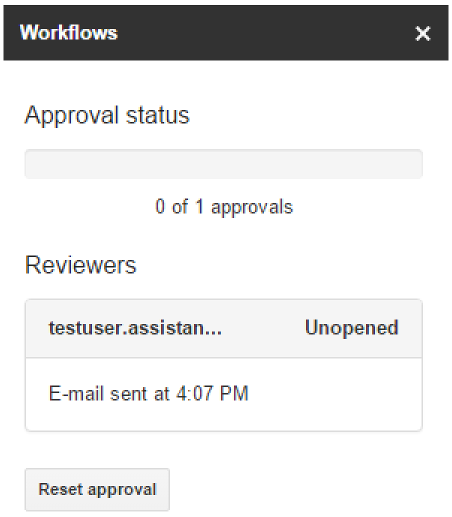
check out this amazing document

Review

Include a comment (optional).

(b) Request for approval for reviewer

Figure 6.1: Letter Feed Workflow example


Workflows

Approval status

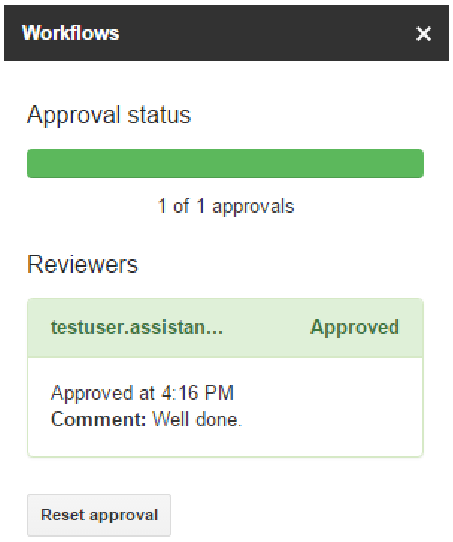
0 of 1 approvals

Reviewers

testuser.assistan...	Unopened

E-mail sent at 4:07 PM

(a) Approval status for owner


Workflows

Approval status

1 of 1 approvals

Reviewers

testuser.assistan...	Approved

Approved at 4:16 PM
Comment: Well done.

(b) Approved status for owner

Figure 6.2: Letter Feed Workflow example: Approval status

the data flow and the interaction is task-centric, like the human-centric approaches. The presented approach does not handle communication outside of tasks either, but within a task, there is the possibility to comment on the document or start a chat.

Similar to the task-centric approaches, a lot of power lies in the hands of the users but contrary to them, the CollaborationScript allows the interaction of more than one user on a task.

With regards to artifact-centric approaches, the CollaborationScript also can restrict access to artifacts but it furthermore supports predefined states for the status. hADL identifies different work-centric human components and coordination-centric connectors, likewise the CollaborationScript can identify human actors and the business process engine. In contrast to ad-hoc processes driven by documents, the CollaborationScript applied to Google Docs provides commenting and chatting.

6.2 Discussion of open issues

As every software faces limitations, the presented implementation is not the exception. Thus, to keep within bounds of a Bachelor thesis some restrictions have been undertaken.

As a first constraint, the "Retracted" step of the state machine of Little-JIL was not implemented. As mentioned in the Chapter Methodology the integration of this step was not possible due to the missing connection to a business process engine. Future work that integrates this functionality will need to make use of a callback mechanism, permanently checking for retraction done by the business process engine.

Another concern of this implementation is the one to one multiplicity of artifacts and tasks. Expanding this restriction to a one to many relationship between tasks and documents involves further user interface elements. Therefore, the challenge will be to make it absolutely clear to the user which document is assigned to which task.

Furthermore, the script bound to the document could be published as an Google Add-On to make it available in the Google Docs add-on store, such as the Letter Feed Workflow example. Doing so, would entail an extension of the implementation with the possibility to define users, roles, permissions, tasks - in short all needed data - within the frame of Google Docs.

As has been mentioned shortly, the script could be easily adapted to Google Docs, Spreadsheets or even Forms. This would only need the change of some function calls in the script, namely `DocumentApp` to `SpreadsheetApp` or `FormApp`.

Finally, another limitation lies in the definition of the data. All the used data, such as user, roles, permissions, and task were mock data written down in a JSON data structure. In future, this data should be defined outside of Google Apps and can be easily delivered to the Configuration Service, as this already provides the necessary interfaces.

CHAPTER 7

Summary

In this thesis an overall architecture for collaboration applicable to different frameworks was developed. As highlighted in the problem statement and the motivating scenario, the architecture should solve the problem of workflow management with status for tasks, role defining and interfaces to make it expandable and able to interact with a business process engine.

The state of the art chapter showed different approaches to model processes with a special focus on support for human interaction and communication outside of tasks. Subsequently, the implementation, the main part of this thesis, was explained, containing its specific architecture with its components, methods and an explanation of how to integrate it into Google Docs. In the evaluation chapter a sample use case that applies the architecture to Google Docs was shown. Finally, the implementation and architecture was critically reflected and the open issues were outlined.

From an engineering point of view, interesting future work includes the integration of a business process engine. Such work could pick up existing methods and advance the current framework. This would provide novel possibilities to expand the workflow.

Bibliography

[1] Abecker, A., Bernardi, A., Maus, H., Sintek, M., and Wenzel, C. Information supply for business processes: coupling workflow with document analysis and information retrieval. *Knowledge-based systems*, 13(5):271–284, 2000.

[2] Bernstein, A. How can cooperative work tools support dynamic group process? bridging the specificity frontier. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, CSCW '00, pages 279–288, New York, NY, USA, 2000. ACM.

[3] Bhattacharya, K., Hull, R., Su, J., et al. A data-centric design methodology for business processes. *Handbook of Research on Business Process Modeling*, pages 503–531, 2009.

[4] Börger, E. Approaches to modeling business processes: a critical analysis of bpmn, workflow patterns and yawl. *Software & Systems Modeling*, 11(3):305–318, 2012.

[5] Brambilla, M., Fraternali, P., and Vaca, C. Bpmn and design patterns for engineering social bpm solutions. In Daniel, F., Barkaoui, K., and Dustdar, S., editors, *Business Process Management Workshops*, volume 99 of *Lecture Notes in Business Information Processing*, pages 219–230. Springer Berlin Heidelberg, 2012.

[6] camunda services GmbH. User task. `https://docs.camunda.org/manual/7.4/reference/bpmn20/tasks/user-task/`. Accessed: 01-03-2016.

[7] Cass, A. G., Lerner, B. S., McCall, E. K., Osterweil, L. J., Sutton, M. S. J., and Wise, A. Little-jil/juliette: A process definition language and interpreter. *ICSE*, pages 754–757, 2000.

[8] Dengler, F., Koschmider, A., Oberweis, A., and Zhang, H. Social software for coordination of collaborative process activities. In *Business Process Management Workshops*, pages 396–407. Springer, 2011.

[9] Ditze, A. and Henninger, T. Methodological approach for business analysts with bpmn. *Modeling Magazine Issue 5*, 2010.

[10] Dorn, C. Collaboration pattern modeling in support of norm specification, monitoring, and preservation. 2015.

[11] Dorn, C. and Dustdar, S. Supporting dynamic, people-driven processes through self-learning of message flows. In *Advanced Information Systems Engineering*, pages 657–671. Springer, 2011.

[12] Dorn, C., Dustdar, S., and Osterweil, L. J. Specifying flexible human behavior in interaction-intensive process environments. In Sadiq, S., Soffer, P., and Völzer, H., editors, *Business Process Management*, volume 8659 of *Lecture Notes in Computer Science*, pages 366–373. Springer International Publishing, 2014.

[13] Dorn, C., Marín, C. A., Mehandjiev, N., and Dustdar, S. Self-learning predictor aggregation for the evolution of people-driven ad-hoc processes. In *Business Process Management*, pages 215–230. Springer, 2011.

[14] Dorn, C. and Taylor, R. Architecture-driven modeling of adaptive collaboration structures in large-scale social web applications. In Wang, X., Cruz, I., Delis, A., and Huang, G., editors, *Web Information Systems Engineering - WISE 2012*, volume 7651 of *Lecture Notes in Computer Science*, pages 143–156. Springer Berlin Heidelberg, 2012.

[15] Dustdar, S. Caramba—a process-aware collaboration system supporting ad hoc and collaborative processes in virtual teams. *Distributed and Parallel Databases*, 15(1):45–66, 2004.

[16] Erol, S., Granitzer, M., Happ, S., Jantunen, S., Jennings, B., Johannesson, P., Koschmider, A., Nurcan, S., Rossi, D., and Schmidt, R. Combining bpm and social software: contradiction or chance? *Journal of software maintenance and evolution: research and practice*, 22(6-7):449–476, 2010.

[17] Fleischmann, A., Schmidt, W., Stary, C., Obermeier, S., and Brger, E. *Subject-oriented business process management.* Springer Publishing Company, Incorporated, 2012.

[18] Fraternali, P., Brambilla, M., and Vaca, C. A model-driven approach to social bpm applications. *Social BPM. Future Strategies Inc.(May 2011)*, 2011.

[19] Gaaloul, K., Charoy, F., and Schaad, A. Modelling task delegation for human-centric egovernment workflows. In *Proceedings of the 10th Annual International Conference on Digital Government Research: Social Networks: Making Connections Between Citizens, Data and Government*, dg.o '09, pages 79–87. Digital Government Society of North America, 2009.

[20] Group, L. P. W. et al. Little-jil 1.5 language report. *Laboratory for Advanced Software Engineering Research, Dept. of Computer Science, Univ. of Massachusetts-Amherst, MA*, 2006.

[21] Grünert, D., Brucker-Kley, E., and Keller, T. obpm–an opportunistic approach to business process modeling and execution. In *Business Process Management Workshops*, pages 463–474. Springer, 2014.

[22] Hilton, P. camunda developer-friendly bpm. `http://camunda.com/bpm/features/`. Accessed: 01-03-2016.

[23] Ings, D., Clément, L., König, D., Mehta, V., Mueller, R., Rangaswamy, R., Rowley, M., and Trickovic, I. Ws-bpel extension for people (bpel4people) specification version 1.1. *OASIS Committee Specification (August 2010)*, 2010.

[24] Ings, D. et al. Web services–human task (ws-humantask) specification version 1.1. *OASIS Committee Specification (August 2010).*

[25] Künzle, V. and Reichert, M. Philharmonicflows: towards a framework for object-aware process management. *Journal of Software Maintenance and Evolution: Research and Practice*, 23(4):205–244, 2011.

[26] Kuo, J.-Y. A document-driven agent-based approach for business processes management. *Information and Software Technology*, 46(6):373–382, 2004.

[27] La Rosa, M., Dumas, M., ter Hofstede, A., Mendling, J., and Gottschalk, F. Beyond control-flow: Extending business process configuration to roles and objects. In *Conceptual Modeling-ER 2008*, pages 199–215. Springer, 2008.

[28] Liptchinsky, V., Khazankin, R., Truong, H.-L., and Dustdar, S. A novel approach to modeling context-aware and social collaboration processes. In *Advanced Information Systems Engineering*, pages 565–580. Springer, 2012.

[29] Malone, T. W., Crowston, K., and Herman, G. A. *Organizing business knowledge: the MIT process handbook.* MIT press, 2003.

[30] Meyer, A. and Weske, M. Activity-centric and artifact-centric process model roundtrip. In Lohmann, N., Song, M., and Wohed, P., editors, *Business Process Management Workshops*, volume 171 of *Lecture Notes in Business Information Processing*, pages 167–181. Springer International Publishing, 2014.

[31] Müller, D., Reichert, M., and Herbst, J. Data-driven modeling and coordination of large process structures. In Meersman, R. and Tari, Z., editors, *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS*, volume 4803 of *Lecture Notes in Computer Science*, pages 131–149. Springer Berlin Heidelberg, 2007.

[32] Moody, P., Gruen, D., Muller, M. J., Tang, J., and Moran, T. P. Business activity patterns: A new model for collaborative business applications. *IBM Syst. J.*, 45(4):683–694, October 2006.

[33] Nandi, P., Koenig, D., Moser, S., Hull, R., Klicnik, V., Claussen, S., Kloppman, M., and Vergo, J. Data4bpm, part 1: Introducing business entities and the business entity definition language (BEDL). `http://ibm.co/1QTR0IH`. (April 2010).

[34] Redding, G., Dumas, M., ter Hofstede, A. H., and Iordachescu, A. A flexible, object-centric approach for business process modelling. *Service Oriented Computing and Applications*, 4(3):191–201, 2010.

[35] Sungur, C. T., Dorn, C., Dustdar, S., and Leymann, F. Transforming collaboration structures into deployable informal processes. In Cimiano, P., Frasincar, F., Houben, G.-J., and Schwabe, D., editors, *Engineering the Web in the Big Data Era*, volume 9114 of *Lecture Notes in Computer Science*, pages 231–250. Springer International Publishing, 2015.

[36] Touzi, J., Benaben, F., Pingaud, H., and Lorré, J. P. A model-driven approach for collaborative service-oriented architecture design. *International Journal of Production Economics*, 121(1):5–20, 2009.

[37] Turner, M. and White, S. A. Software development process. `http://sce.uhcl.edu/whiteta/sdp/softwareDevelopmentProcess.html`. Accessed: 01-04-2016.

[38] Wang, J. and Kumar, A. A framework for document-driven workflow systems. In *Business Process Management*, pages 285–301. Springer, 2005.

[39] Weske, M. Flexible modeling and execution of workflow activities. In *System Sciences, 1998., Proceedings of the Thirty-First Hawaii International Conference on*, volume 7, pages 713–722 vol.7, Jan 1998.

[40] White, S. A. Introduction to bpmn. *IBM Cooperation*, 2(0):0, 2004.