

A Mashup Tool for Collaborative Engineering of Service-oriented Enterprise Documents

Nelly Schuster¹, Raffael Stein², Christian Zirpins²,

¹ FZI Forschungszentrum Informatik, Haid-und-Neu-Straße 10-14,
76131 Karlsruhe Germany,
{nschust}@fzi.de

² Karlsruhe Institute of Technology (KIT),
76131 Karlsruhe Germany,
{raffael.stein, christian.zirpins}@kit.edu

Abstract. Enterprise documents combine the representation of organizational processes and rules with knowledge and data to support human communication in a visual appealing and possibly interactive way. These characteristics are not only beneficial for the support of highly structured and optimized business processes but can be also leveraged to drive ad hoc collaborative innovation processes. However, collaborative authoring requires content consolidation and coordination mechanisms and while IT-supported document engineering for structured and recurring collaboration processes is well established, engineering of documents that emerge and evolve instantaneously still lacks appropriate support. Subsequently we present the MoSaiC tool for collaborative document engineering: MoSaiC supports a new type of service-oriented enterprise documents that are represented as mashups of content creation, transformation and publication services. This ‘living’ format supports teams to rapidly regulate and control collaborative activities by mapping them to services provided by team members or enterprise systems and mashing them into document structure and rules in an interactive, intuitive and dynamic way.

Keywords: service-oriented enterprise documents, collaborative document engineering, situational document collaboration

1 Introduction

Enterprise documents provide a unique way to communicate information in a purpose-optimized (structured, annotated, graphically appealing, legally binding) form of representation in order to collect and share data between human recipients in the course of collaborative work. Sophisticated enterprise documents encapsulate business rules and serve as input to or output of business processes, thus enabling their regulation and enforcement in a highly efficient manner; e.g. utilizing customer relationship, and workflow management systems (WfMS). While several enterprise document engineering approaches exist for structured and recurring business processes [1,2], the engineering of documents which evolve during ad hoc collaboration between members of a possibly virtual team still lacks appropriate tool support. Such

documents are subject to ad hoc changes as new contents from different sources – humans and software systems – come in. Dependencies between different parts of a document require coordination mechanisms and ad hoc process support for the authors of the document. Examples can be found for instance in IT service management or software engineering, where collaborators have to react on unexpected incidents like bugs or are collaboratively developing solutions which are captured in interrelated documents [3]. Another example is the collaborative development of a research publication [4]. Various technologies and tools support either collaborative creation and evolution of documents (e.g. CSCW technologies, wikis or other Web-based collaborative writing applications [5]) or coordination of structured processes (e.g. BPM technologies like WfMS). However, they largely fail to provide effective support for collaborative document engineering which involves ad hoc processes.

In order to address this need, we have developed the MoSaiC approach for service-oriented enterprise document engineering. We model enterprise documents as situational compositions (so called mashups) of human- or software-based content provision, transformation or publication services provided by human collaborators or Web-based software services [6]. Lightweight coordination of collaborating authors and auxiliary Web services is facilitated by interaction rules that are defined as part of the document in a declarative way. Thus, a document is ‘alive’, reacting to changes of the document itself and its underlying services.

In this paper, we present a Web-based mashup tool and platform that implement our collaborative document engineering approach. In Chapter 2 we will sketch the concept of collaborative document mashups. Chapter 3 outlines the architecture of our mashup tool and platform. An exemplary use case is given in Chapter 4. We discuss related work in Chapter 5. Chapter 6 concludes with a summary and outlook.

2 Concept of Collaborative Document Mashups

Collaborative document engineering requires coordination of team members that jointly work on and evolve ad hoc documents as well as consolidation of content coming from a variety of sources including team members but also data sources. Content consolidation in document mashups is achieved through representing manual as well as automated activities through software services that are composed into the document structure. Document mashups build on a hierarchical structure that represents the logical decomposition of a collaboration goal into tasks. This structure is defined by one or more coordinating team members (*coordinators*). Subsequently, tasks are bound to document services representing either collaborative activities of team members (*collaborators*) or automated functions of software systems (*robots*) that result in creation (e.g., writing text, drawing diagrams, accessing databases) or transformation (e.g., proof-reading, translating, layout) of document contents like chapters, paragraphs or figures. As soon as a service is bound to a document mashup the service provider becomes a participant (either co-author or robot) of the document collaboration responsible for the associated task.

The coordination of service providers (i.e. collaborators) in an evolving document mashup builds on events and rules: each service exposes events indicating state transi-

tions of its underlying resources (i.e. activity results) which are consumed and reacted upon by document mashups. An important part of document engineering is to define ECA rules that are triggered by events and lead to corresponding actions thus expressing causal dependencies or temporal constraints between service instances; e.g., a rule may specify that a proof-read service is called by the document mashup as soon as a text-based content service emits an update event. Another example might be the recall of all document services that failed to deliver results at a certain deadline. Coordinators are free to specify any rules that fit the collaboration at hand.

We characterize document service mashups as ‘living’ because they are constantly evolving in terms of structure and contents towards achieving a common goal. To a large extent this dynamicity is driven by manual interaction of team members with the document: Document engineers evolve and control the document structure and rules. Collaborators create or transform document contents.

Additionally, certain aspects of document evolution are controlled by rules that react on events and trigger activities of team members and data sources automatically. Rules of a document mashup will immediately be executed when the event they listen for is emitted. This might lead to further activities like service calls.

During the course of the collaboration, document content and structure transform from a record of collaborative conversation to a meaningful result. As we do not separate phases of design and enforcement, service calls will be made as soon as services are added to a document mashup and each change of the document structure immediately effects the control of a collaboration. Still, in order to enable the design of critical document parts, mashups can be set ‘asleep’. This means, that the control of document services is idle until the mashup is ‘woken up’ again.

3 MoSaiC Tool and Platform Architecture

In order to realize our concept of document service mashups we have developed a Web-based platform and RIA to support collaborative document engineering. Figure 1 gives an overview of the architecture.

Users access the document collaboration environment through a graphical user interface that runs on their Web browser. The UI communicates with the application and persistence logic of the MoSaiC document mashup manager which might run on an enterprise application server or in the cloud. The GUI uses AJAX calls to communicate with the server components. The collaboration system uses a communication infrastructure for event-based interaction between services in document mashups. This infrastructure – the document service bus – might be deployed independently and is accessed through RESTful service interfaces.

The Web-based document collaboration tool integrates various graphical drag-and-drop user interface components supporting mashup authoring, service provisioning and participant management use cases:

- Fundamentally, the *user management UI* allows logging into the system or registering as a new user. Subsequently, the *participant UI* enables users to specify their role in one or more document collaborations. This might include the responsibility of a coordinator to participate in the engineering of the document itself. This might

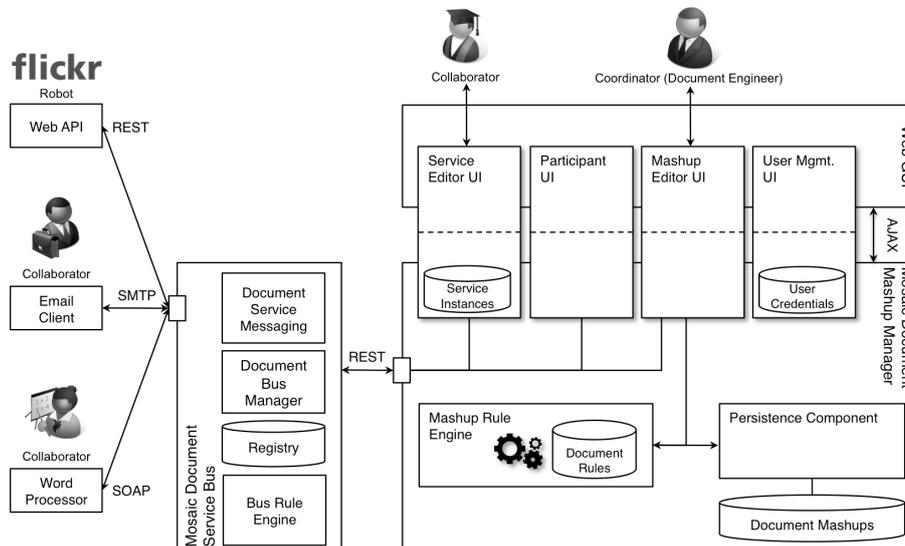


Figure 1: Architectural Overview of the MoSaiC Mashup Tool and Platform

also include the responsibility of a collaborator to carry out certain activities within a collaboration and provide associated document services manually. As a variant, users might delegate responsibility to a robot that provides document services by means of automated Web services (e.g. to integrate contents from Flickr).

- Using the *mashup editor UI*, coordinators carry out the main document engineering tasks including design and management of collaborative document mashups. Management includes administrating a repository of living documents. It also includes state control of individual mashups which happens in sync with an ongoing design process. To this end the editor represents and visualizes document mashups from different perspectives of structure, collaboration and coordination. In the structure perspective, engineers design the logical document structure on a canvas, e.g. a research paper split into parts of the research methodology as well as result tables and diagrams. In the collaboration perspective, the UI shows lists of content and transformation services as well as possible providers. The coordinator can drag and drop the services from this list on the canvas or assign providers to elements. The coordination perspective allows definition and management of rules expressing causal and temporal constraints as well as manual control of actions.
- In order to support collaborators of a document mashup in providing contributions through service calls, the platform offers a *service editor UI*. This UI allows users to receive service calls coming from a document mashup manager (the same or any other), edit and persist appropriate content, e.g. the introduction of a research paper, and communicate it back to the document mashup manager. However, there are also other ways for collaborators to connect like an email bridge or plug-ins for common word processors that enable them to handle document service calls.

Mashup logic is enforced by the mashup manager. Its *persistence component* stores all mashups including structure, tasks, rules, services, participants, roles, contents and control state. The persistence component listens to mashup and service update events

and stores the information accordingly. The UI components learn about updates of a mashup through polling the persistence component.

A *mashup rule engine* drives automatic enforcement of mashup logic. It is able to process complex events and pre-configured with base rules, which are essential for the enforcement of mashups. Rules follow the event-condition-action (ECA) principle: the engine listens for events indicating changes of the mashup or its services, checks conditions and takes actions like calling another service. E.g. one of the base rules declares that if a user gets assigned as collaborator, a 'create' call is sent out to its document service. In addition, coordinators specify rules that are mashup-specific. E.g. the above base rule might be changed to express a causal dependency.

A *document service bus* enables service interaction. Fundamentally, the bus maintains a *registry*, which stores information about existing mashups, document services and providers. The *document service messaging* component routes and queues all service messages. Service calls generally conform to a uniform RESTful interface and an asynchronous interaction protocol that allows requesting the creation of a new service as well as getting, updating or deleting the content of an existing one. Furthermore, a dedicated *bus rule engine* allows for a specific type of rules that directly affect the interaction between services. Enforcing rules in the bus promises more agile interactions since messages skip the mashup manager if possible. An example is the automatic routing of certain text content through a translation service before delivering it to the mashup manager.

4 Scenario and Use Case

Scientific publication involves collaboration of various researchers who provide different parts of a paper like texts, pictures and references. Furthermore the collaborators need to coordinate in order to discuss the concepts, structure the document, provide interrelated contents and proof-read the results thereby checking for completeness, correctness and readability.

In our scenario, Alice, a scientist, is creating a research paper mashup and starts with defining the structure of the document in the mashup editor. She drags several document elements for different parts, texts and figures from a list to the mashup canvas. Furthermore, she adds elements for an abstract and bibliography. For each element Alice provides a meaningful task description and whenever she drops an element on the canvas, the change in the mashup structure is immediately persisted.

Having established the first structural draft, Alice defines behavioral rules in the coordination perspective. She creates a rule declaring that all chapters need to be delivered three days before the conference deadline; i.e. each of the chapters has been updated and its state set to final. A timer is defined, which triggers a notification to all the authors who did not deliver content until the specified point of time. All rules are stored within the mashup and within the rule engine.

In the document service list, Alice identifies a layout service that is able to format mashups into an appropriate format and a submission service that uploads a formatted document to the conference server. For both services there is a robot that she can add to the mashup. She defines a rule that the mashup is formatted by the layout service

when all content elements are in final state. Afterwards a final proof-read by the first author is required and, in case the proof-read is ok, the submission service uploads the document to the conference server.

Having specified the rules, Alice uses the provider list in order to find collaborators that can assist with the tasks at hand. She discovers Bob, Carol and Ted, who are members of her research team and have already agreed to join the paper. Alice drags the icon for Bob on the introduction document part; the association between him and a document service representing the writing task is immediately stored. Bob uses the same tool to provide his content as a document service. Now Alice wakes up the mashup. Bob receives a notification that he has been assigned to write this paragraph. The task is added to his personal to-do list, which is presented to him in the service editor UI after logging into the mashup tool.

In the following weeks, the team does several changes to the initial mashup to evolve the paper in terms of structure, tasks and, of course, contents. Three days before the deadline the rule engine triggers the “reminder” rule. It identifies the document elements which are not in final state and triggers updates of the associated services. Since Bob did not deliver yet, he receives another service call for his document service to provide the text. He logs into the system, pastes the introduction text into the service editor, refines the document element adding a picture element and marks its state as final.

The rule engine observes the state change and triggers another rule, which declares that as soon as all section elements are in final state, the mashup is to be sent to the layout service. After the layout service returned a formatted document, the first author is requested to proof-read. A reply indicating success triggers the final action: the formatted document is sent out by the submission service.

5 Related Work

MoSaiC adopts concepts of collaborative document engineering and writing based on service mashups for ad hoc composition of mostly human-based software services.

Collaborative document engineering has a long research history, which resulted in various research prototypes and products. A prominent example is Google Docs (<http://docs.google.com/>) that allows collaborative creation of rich text documents. However, it is not possible to define any dependencies between document elements or react to events. Thus, lightweight ad hoc processes cannot be specified or enforced. Another related technology coming closer to our approach is Google Wave (<http://wave.google.com/>). A wave is a collaboration of participants based on XML documents consisting of wavelets. This is similar to the composition of document services in a mashup. Waves might include automated robots that are comparable to our automated document services. However, Google Wave focuses more on communication than on collaborative evolution of an enterprise document. Also, there is no way to define interaction rules based on events and to re-use wavelets in other waves.

A study of tools for collaborative writing, including early Wiki software, is presented in [5]. The study shows, that the idea of splitting a document into fragments for different authors and propagating updates of these fragments to other authors is not

new. However, we did not find a tool which supports coordination of the authors or fragments through rules or the integration of different sources from the Web.

The term document engineering as used in [1] describes the analysis and design of documents and rules which are input to business processes or serve as their interfaces. This has similarities to our approach but focuses on structured, recurring processes. There is no concept of document composition like in document mashups that would support collaborative or active documents and ad hoc processes. Interactive Web documents [7] define a REST protocol and format for Web based documents which include data and behavior. However, although they mention a prototype, they do not show how these documents can be collaboratively authored and coordinated.

Mashup technologies have recently gained broad attention from industry [8] and are increasingly addressed by academic research. The mashup paradigm emphasizes user-driven composition of situational apps from Web-based content and services. Several mashup research tools and products exist. An example is IBM Mashup Center (<http://www-01.ibm.com/software/info/mashup-center/>) that lets users compose widgets which reference services. Based on the description which events a widget might expose, a mashup developer can add rules which route data between widgets. However, since these events are mainly caused by user clicks and not by updating any content and as they are exposed by widgets and not services or the mashup, it is not possible to specify complex rules involving several events or conditions. Collaboration support is also limited, since only the full mashup can be updated at a time. Also, the mashup is still an application and cannot be used as an active document.

Loomp [9] allows users to specify content fragments and enrich them semantically. These reusable fragments are readable by humans and machines alike and might be interlinked and combined into mashups. Distribution of fragments or data can be done through various channels, for instance as Wiki page, document or feed. However, there seems to be no support for notification on updates of fragments or process support for collaborative document engineering.

6 Summary and Outlook

Shaping and driving ad hoc collaboration processes by means of enterprise document engineering promises to enhance productivity and foster innovation but also poses substantial requirements on flexibility of document structure and dynamicity of contents. Respective documents need to reflect progressing states of multiple collaborators and render their visual representation from various sources. Simultaneously, collaborators need reliable tools allowing them to evolve the structure of collaborative documents in highly interactive yet predictable ways. Our approach of document service mashups establishes a basis to meet these requirements that we would like to discuss and share with the community. Document mashups support interactive specification of the structure and collaborative regulation of a shared document in a flexible declarative way on the fly. Document services facilitate the integration of dynamic content from various collaborators.

In this paper, we have presented a mashup tool and platform for collaborative engineering of service-oriented enterprise documents. We have briefly outlined our

approach of collaborative document service mashups and demonstrated how to realize it by means of contemporary service-oriented infrastructure technologies including RESTful Web services, complex event processing and rich Web 2.0 Internet applications. Aiming for a practical perspective to present our work, we have illustrated our general approach and the operation of our tool by means of a case study coming from the familiar scenario of scientific collaboration.

For future work we plan to underpin and evolve the current basis of collaborative document service mashups in various directions. At the moment we are thoroughly evaluating our prototype platform by means of case study experiments. Simultaneously we are extending the mashup engineering methodology by formal verification of rule declarations and service value network analysis. We further plan to extend document mashups with versioning information that enable traceability of changes and lead to more transparency. Finally, we aim to study document interaction patterns for common control structures (like, e.g., the ‘four-eye principle’) for the specific case of situational document collaboration. We intend to provide our findings as reusable design patterns for ad hoc development of document mashups.

References

1. Glushko, R. J. and McGrath, T.: Document engineering: analyzing and designing documents for business informatics & web services. The MIT Press (2008)
2. Hull, R.: Artifact-Centric Business Process Models: Brief Survey of Research Results and Challenges. In: OTM Conferences 2, Vol. 5332, pp. 1152-1163. Springer (2008)
3. Schuster, N., Zirpins, C., Tai, S., Battle, S., and Heuer, N.: A service-oriented approach to document-centric situational collaboration processes. In: 18th IEEE Int. Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises, pp. 221–226, IEEE Comp. Society (2009)
4. Marchese, M., Giunchiglia, F., and Casati, F.: Liquid publications: Scientific publications meet the web. Technical report DIT-07-073, University of Trento, Department of Information Engineering and Computer Science (2007)
5. Noël, S., and Robert, J.-M.: How the Web is used to support collaborative writing. In: Behaviour & IT, Vol. 22, Nr. 4, pp. 245-262 (2003)
6. Schuster, N., Zirpins, C., Schwuchow, M., Battle, S., and Tai, S.: The MoSaiC Model and Architecture for Service-Oriented Enterprise Document Mashups. In: 3rd International Workshop on Web APIs and Services Mashups (Mashups'09), OOPSLA, ACM (2009)
7. Boyer, J. M., Wiecha, C., and Akolkar, R. P.: A REST protocol and composite format for interactive web documents. In: ACM Symposium on Document Engineering ACM, pp. 139-148 (2009)
8. Hoyer, V. and Fischer, M.: Market overview of enterprise mashup tools. In: ICSOC '08: Proceedings of the 6th International Conference on Service-Oriented Computing Berlin, pp. 708-721. Springer, Heidelberg (2008)
9. Luczak-Roesch, M., and Heese, R.: Linked Data Authoring for Non-Experts. In: Proceedings of the WWW09, Workshop Linked Data on the Web LDOW2009 (2009)