Lightweight Collaboration Management

Matthias Kunze   Alexander Grosskopf
Hagen Overdick   Matthias Weidlich

Hasso-Plattner-Institute at the University of Potsdam
Prof.-Dr.-Helmert-Str. 2-3
14482 Potsdam, Germany
{matthias.kunze, alexander.grosskopf, hagen.overdick, matthias.weidlich}@hpi.uni-potsdam.de

ABSTRACT

Collaboration processes are generally coordinated without an explicit notion of a guiding process. Even though this kind of work is performed in a rather structured manner, explicit software support to coordinate these processes is rare. On the other hand, process automation is mainly considered for highly frequent processes, due to the cumbersome setup of adequate systems and the process implementation effort.

This paper presents a mashup that effectively coordinates humans who strive for a collaborative goal. Participants can design and enact their processes right away; a lightweight process execution engine automatically coordinates participants through correlated messages. In contrast to classic mashups, we turn the architecture upside down and orchestrate Web applications and their respective service APIs. A process map, similar to classic mapping mashups, gives insight into the current state of a process and its activities as well as information that is required to evaluate and trace process history.

Categories and Subject Descriptors

D.2.11 [**SOFTWARE ENGINEERING**]: Software Architectures—*Patterns*; H.3.5 [**INFORMATION STORAGE AND RETRIEVAL**]: Online Information Services—*Web-based services*; H.4.1 [**INFORMATION SYSTEMS APPLICATIONS**]: Office Automation—*Workflow management*; H.5.3 [**INFORMATION INTERFACES AND PRESENTATION**]: Group and Organization Interfaces—*Collaborative computing*

General Terms

Design, Experimentation, Human Factors

1. INTRODUCTION

The term Web 2.0, often referring to a set of technologies comprising an architecture of participation [30], does also relate to a changed perception of the Web—a paradigm shift from consumption to participation. On the one hand, people are connecting through social networks, exposing their social and professional life in weblogs, and share their knowledge in wikis. On the other hand, people located at distant places collaborate to achieve a common goal. As a result, the role of message exchanges changed dramatically. Messages are no longer used solely for knowledge exchange. Instead, they are often used to coordinate people that work together.

The majority of these collaborative efforts are, often unconsciously, conducted in a rather structured manner; they comply with previously agreed guidelines or business rules. In many cases they involve a certain order of tasks that are dedicated to particular participants. Consequently, these efforts can be regarded as processes: a set of activities, performed in coordination in an organizational and technical environment, that realize a particular goal [39].

The area of Business Process Management addresses the explicit support of processes on operational and organizational levels. Mature solutions exist to automate complex and highly frequent processes in very efficient ways. However, most processes are accomplished by persons and not by workflow engines. Such processes naturally have a small frequency, setting them up formally is considered cumbersome and not worthwhile.

In this paper, we identify and analyze the challenges of collaboration management by means of business processes and elaborate on a software solution to support these processes and guide participants through collaborative processes. We argue that the common way of organizing a collaboration in an ad-hoc way has various drawbacks, such as a lack of consistency, transparency, and traceability. However, full-fledged collaboration management systems often imply immense configuration overhead and do not integrate unobtrusively with existing work places of collaboration participants.

Therefore, we propose to combine the concepts of process-driven collaboration management with the mashup paradigm. Typically, mashups are a lightweight combination of existing information into a new view and "save valuable time, since all data can be put in one page" [32]. The main benefit of mashups is considered to gain insight among information that originates from disparate sources [17, 19, 43]. For the setting of process-driven collaboration management, we leverage the very same concepts and technologies mashups use for lightweight application composition, but turn the classic

model upside down. Instead of aggregating disparate services into one, we orchestrate Web applications and their respective service APIs behind the scenes and retain their interface to users. That, in turn, allows for lightweight process-driven collaboration management. In order to validate our concept, we implemented a research prototype. We also report on findings we made, using the prototype in the course of writing this paper.

The remainder of this paper is structured as follows. Section 2 discusses the drawbacks of organizing collaboration in an ad-hoc fashion and derives requirements for information systems aiming at support for collaboration processes. Based thereon, we propose an architectural style in Section 3 and compare it to known architectural styles. In Section 4, we present our prototype and the lessons learned from implementing and using the solution. Finally, we review related work in Section 5 and conclude the paper in Section 6.

2. COLLABORATION MANAGEMENT

While there is no doubt about the importance of Business Process Management in the context of highly automated processes, human-centered processes are often neglected. Although there are a few drivers that lead to the creation of process models for human-centered processes, among them staff planning, employee training, and business certification, the way people conduct processes in their everyday life is rarely considered. In particular, it is a common observation that collaboration processes are coordinated without an explicit notion of a process, even though this kind of work is performed in a rather structured manner.

Focusing on these implicit collaboration processes as our use case, we first shortly sketch how these processes are realized in Section 2.1. Based thereon, we derive requirements for information systems that aim at supporting collaboration processes with an explicit process representation in Section 2.2.

2.1 Realization of Collaboration Management

In order to illustrate the major activities of a collaboration process, we use the process of jointly writing a scientific paper as a well-known collaboration example.

Setup & Enact. Usually, collaboration is initiated by agreeing on a roadmap in order to reach a certain goal. Further on, tasks are assigned to participants. Often the enactment of collaboration is manifested in a kick-off meeting, while task assignments are captured in meeting protocols that are distributed via e-mail or wiki pages. Enactment also involves the setup of the infrastructure that is needed for further collaboration. In case of writing a scientific paper, the process is initiated by a meeting in which the authors team up, agree on a submission target, and assign work packages. The latter includes the duty to write a certain paper section, to prepare and conduct experiments, or to implement certain algorithms. In addition, a version control repository is created and all authors are provided with access.

Communicate & Coordinate. During the collaboration, people interact with the help of short messages: e-mail, instant messaging, and recently also through micro blogging such as Twitter. If one participant finishes a task, they signal this to participants responsible for subsequent tasks and thus, coordinate the work based on the roadmap. With respect to the exemplary process, in this phase an author notifies the co-authors that they committed a new section in the paper and request feedback. Further on, issues that come up during an implementation of the theoretical concepts of the paper are discussed via email. Based on the progress in the repository, the coordinating author (often, the first author) contacts their co-authors, such that they meet the deadlines as agreed.

Trace & Evaluate. Evidently, coordination of the collaboration assumes an overview of the current state of the process in order to ensure compliance with the roadmap as agreed. In addition, participants want to know when they can expect their next task to manage their workload. That, in turn, involves various interactions using short messages. In case of the paper writing scenario, the history of exchanged messages between the authors as well as the commit log of the repository can be exploited for the purpose of traceability and evaluation. For instance, the design decisions that have been taken in order to come up with a certain architecture for the implementation can be reconstructed. Further on, information of the commit log can be leveraged for improving time estimations when writing the next paper.

It is worth to mention that we do not consider these activities to be done in a linear order. In particular, coordination and evaluation are tightly coupled, as coordination might be based on the evaluation of the progress for obvious reasons.

The aforementioned realization of a collaboration process in an implicit fashion reveals several drawbacks.

Inefficient. The implicit decentralized coordination of participants breaks the flow of work. That is, a participant might have to wait for the approval of a previous participant, even though the process allows immediate action.

Intransparent. Discussions typically require various interactions leading to *'ping-pong'* processes that are either not externally visible or hard to trace due to the information density and the multitude of used communication channels.

Inconsistent. Inconsistencies between the processing as agreed and the *'natural'* flow of work cannot be detected due to the aforementioned lack of transparency.

Not measurable. The implicit coordination results in the absence of any global progress information that might be measured. Therefore, it is not possible to compare similar collaborations in a reasonable way.

Scattered artifacts. Information artifacts that are used or created in the scope of the collaboration reside at different data stores and are not linked to the message based discussion and process coordination.

2.2 Requirements for Collaboration Support

Addressing the drawbacks of a traditional realization of collaboration processes, we identify the following requirements for information systems that aim at full-fledged support for collaboration processes.
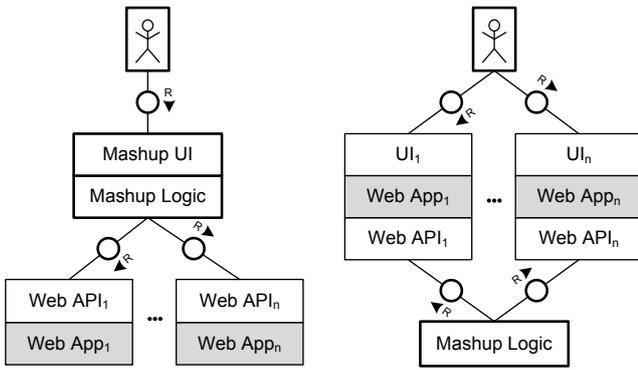
Figure 1: Architectural styles of mashups: classic mashup (left), reverse mashup (right)

Explicit process representation. In order to overcome inefficiency caused by the decentralized coordination, an explicit process representation is needed. Without any formalization of the collaboration roadmap, delays that result from manual coordination cannot be eliminated.

Correlated messages. Traceability of the collaboration requires that all discussions are externally visible. To cope with the information overhead that is implied by such a transparency, message correlation is inevitable. Albeit an essential feature of process execution engines (cf., [5]), correlation that goes beyond simple tracing of email subjects is typically neglected for human-to-human interaction. In particular, that holds for interactions spanning multiple communication channels.

Directed global messages. Of course, coordination requires directed messages between two participants. However, the existence of a direct addressee should not preclude the possibility to trace the message for the remaining participants. Therefore, a mechanism that allows for global tracing (that includes an appropriate search engine) of directed messages is essential. Here, we consider the common usage of Carbon Copies (CC) in emails to be counterproductive as, besides the fact that it is error-prone, it implies a push mechanism, where a pull strategy would be more convenient.

Unobtrusiveness. As mentioned above, various communication channels and tools are already used as part of the collaboration. Therefore, support by information systems should be as unobtrusive as possible. Instead of introducing new tools, existing infrastructure should be leveraged to provide better collaboration support. That, in turn, enables seamless integration into the existing work places of collaboration participants.

System integration. Linking between different sources of collaboration information has to be supported in order to avoid scattered artifacts. Here, linking between information that is purely used for coordination and the actual representation of the collaboration subject is of particular importance.

3. REVERSE MASHUP ARCHITECTURE

Mashups are applications that consume services, i.e., content and functionality, from disparate sources on the Web and aggregate them in new and innovative ways. Mashups draw upon these services and create value by providing immediate solutions to situational needs and insight through connecting related information [35].

Usually, service providers publish APIs as a by-product to a Web application as so-called Web APIs: interfaces that offer a resource via HTTP. Sometimes the providers of these services are not aware of the reuse of the capabilities they offer, e.g., when a mashup scrapes the rendered Web sites to retrieve information that would otherwise be inaccessible [22]. In the context of this work, we assume service providers to explicitly offer Web APIs as an alternative method to access an application's services, regardless of whether these services provide content or functionality.

Much work has been conducted studying mashups and deriving certain characteristics that eventually led to observations and recommendations in the form of reference models and reference architectures for mashups, e.g., [6, 9, 19, 22, 24]. These observations share the same understanding of an architectural style for mashups: Services of heterogeneous type are offered through Web APIs to be aggregated by mashups that act as intermediaries between the user and the services provided. This evolved from the original desire to gain insight among several disparate information sources at a certain point in time, rendering an intermediary application into an aggregator of resources on the Web. Thus, especially data-centric mashups provide realtime information at a single point in time, e.g., [29, 33, 36, 38, 41]. The corresponding architectural style of classic mashups is illustrated in the left part of Figure 1: Mashups encapsulate services and integrate them into a single and unified application, comprising application logic and the user interface.

The present use case—collaboration management—imposes different requirements. The application needs to provide a continuous service, interacting with remote services over a period of time, whereas, nevertheless, reuse of existing resources is pertinent. The additional need to preserve existing user interfaces of services, which people incorporate into their daily work already, led to an architectural style that supports coordination among services rather than aggregating content. These user interfaces may originate from different kinds of clients for existing services, such as desktop clients, Web clients, or mobile clients. Thus, the services become intermediaries between the users and the mashup application logic, turning the classic architecture upside down. This motivated the term *reverse mashup architecture*, which is depicted in the right part of Figure 1.

Mashups corresponding to this architecture integrate different systems seamlessly to support a workflow among involved systems and participants. This architectural style is well known from enterprise application integration [18]. Many systems are orchestrated to realize a higher goal, typically a business process. Specialized solutions, the enterprise services bus [8], evolved to support this integration paradigm. They became complex systems themselves when adding support for enterprise requirements such as adapters for legacy systems, fail-safe message queues, and data mapping tools.

The reverse mashup architecture brings this idea to the Web-environment and leaves behind all the complexity of the heterogeneous enterprise world. Web applications become intermediaries and provide user interfaces to satisfy a particular task, which is, in turn, leveraged by the mashup through the service's Web API to fulfill a higher endeavor. We choose systems by functionality and accessibility from a large set of publicly available services. We enhance the genuine service by integrating information from other systems and services. As all mashups we don't claim to be failsafe. Web standards such as HTTP and JSON ease the burden of integration and thus allow for new lightweight solutions. We can integrate services that offer an API that is specific to the purpose of the service rather than its originally envisioned application scenarios. The REST architectural style addresses these issues by constraining the interface of a service to remain application independent [16].

We encompass the Web application user interfaces and see the corresponding Web APIs not as alternative but complementing interfaces. While participants keep using the user interface provided by the respective Web application, the mashup uses the corresponding Web API to interact with the service. Thus, the mashup plays the role of a coordinator among the services, rather than one of an aggregator of content, which is the case for the majority of mashups.

4. PROTOTYPE

In order to validate our concept of a reverse mashup architecture and to study the guided collaboration of humans through explicit processes, we implemented a research prototype. First, Section 4.1 introduces the prototype following on the major activities of collaboration management as presented in Section 2.1. Second, we report on our experience on using this prototype in Section 4.2.

4.1 Architecture and Implementation

The architecture of the mashup is depicted in Figure 2. It comprises three applications: Oryx, Twitter, and Xenodot. Xenodot provides the implementation platform for our mashup, whereas the first two are accessed in the reverse mashup style.

Setup & Enact. Collaboration management starts with agreeing on a process and its explicit representation that lays the groundwork for its future execution and evaluation. We resorted to **Oryx** [11] as model designer. Oryx is an extensible process modeling platform that allows users to rapidly create process models of various languages on the Web and share them with others. Process languages are implemented as stencil sets that define notation and syntax of a modeling language, i.e., its graphical representation and rules for the correct composition of model elements. As process language for our mashup, we chose a subset of the Business Process Modeling Notation (BPMN) required to form simple yet easily understandable processes [44]. In addition, swimlanes are necessary to define participants responsible for conducting certain tasks.

Figure 3 shows an example process: Labels of the swimlanes refer to the participants responsible for corresponding tasks. In our prototype, these labels equal the screen names of the
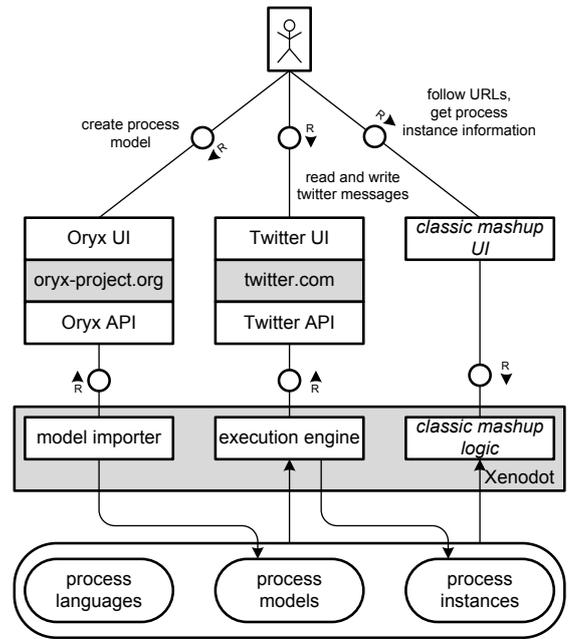

Figure 2: Architecture of the Prototype

participants' Twitter accounts, which we use as communication service, discussed in detail below.

Each process model itself is a Web resource, identified by a URL, its representation being a hypertext document that contains hyperlinks. This allows users to attach any kind of information to a process model or model element. For instance, a jointly written paper could be linked via a URL that points to a document repository.

While Oryx provides its own model repository, we needed a process execution engine that can instantiate processes and coordinate according activities. **Xenodot** is a Web-based process engine that employs the principles imposed by the REST architectural style to store and execute processes. Xenodot uses an HTTP-based document store—Apache CouchDB [3]—to represent elements of process languages, process models, and process instances in a uniform way [31]. Hence, all artifacts in languages, models, and instances are Web resources having their own identity and life cycle and are co-related through URLs.

Process execution with regards to collaboration management is implemented as a specification of operational semantics for the modeling language, i.e., the chosen subset of BPMN. These operational semantics are stored within the language documents within Xenodot: Each model element type has a specific behavior that describes the respective instance's life cycle and actions performed to advance the life cycle's state. At the time of process instantiation, the model and its language's operational semantics are compiled into the process instance's document store as map/reduce functions [10], which in turn are used by the Xenodot execution engine to provide Web-based process enactment.

Communicate & Coordinate. We motivated the use of directed, globally visible messages for participant communication in Section 2.2. **Twitter** is a well suited message service for this purpose: Messages can be directed to particular participants, while they are generally visible to everyone else.[1] Further, through the concept of replying to a previous message, messages can be correlated and discussion graphs can be derived. A simple RESTful API allows sending and retrieving messages. Thus, a robot that is bound to the execution of a process instance could easily communicate through Twitter.

Coordination of participants is realized by a small set of commands that describe the advance of the process' state. A message comprises at least the *addressee* of the message, i.e., its Twitter screen name, the *URL* of a process element, and one *command*. Messages can contain further information that is considered valuable for documentation purposes.

The robot includes the participants' screen names to direct messages to a participant who is responsible for the corresponding action. Participants need to include the robot's screen name in their responses to advance the process' state. At process runtime, the robot requests a participant to conduct a certain task by sending him a `start` message that includes a link to the corresponding process instance activity. When the participant finished this task, they reply with a `done` message containing the same URL. The process engine regularly updates the messages directed to the robot. When it finds a new command, it interprets it and advances the process' state according to the behavior specified in the modeling language. Among others, we distinguish the following command messages.

`@participant <://url> start` is sent by the robot to inform a participant that they are in charge to conduct a certain activity.

`@robot <://url> done` is then replied to indicate that the activity has been finished by a participant.

`@participant <://xor-url> decide` is sent by the robot to ask a participant to choose from several alternative outgoing paths from an exclusive gateway (XOR split).

`@robot <://url> select` is then replied, containing the URL of the first activity of the chosen path.

Each URL in a message aims at the resource of the according process instance element; its representation gives insight about the element's type, name, and current state. It further offers prepared Twitter messages to advance the particular element's state, based on the current state of that element. For example, the representation of an exclusive gateway provides means to choose from a set of alternative outgoing paths.

Trace & Evaluate. In human driven processes, it is important to establish transparency among activities. Participants

need to understand preconditions and impact of their actions to make proper decisions. Thus, advances in the process' state need to be globally visible, cf., Section 2.1.

We approached this requirement by providing a process map in the form of classic mashups: For each process instance, an interactive, visual representation offers an overview of the process' state and related Twitter messages, and allows users to trace actions carried out in the past. If a user clicks on the URL provided with each commanding Twitter message, they will be presented with that map and detailed information about the selected process element. This process map is depicted in Figure 3. Finished activities are highlighted in green, live activities in blue. It is possible to skip activities explicitly for any reason; such activities will be highlighted red. Users can choose any process element in the map to obtain particular information about it.

The same information used to obtain the current state of a process can be used to measure the overall performance of a process. Information stored with each Twitter message allows relating actions that advanced the state of a process to participant roles. Such profiles could then be compared with other artifacts of the collaboration, e.g., document repositories. For example, this allows comparing the amount of time it took to conduct a certain task with the amount of work that has been done. Such information could then be used to refine and improve the process for further application.

4.2   Lessons Learned and Future Work

As mentioned before, we used our research prototype in the course of writing this paper. This section summarizes our experiences in this context.

Implementation. The chosen architecture, introduced in Section 4.1, proved useful during implementation. Adopting the APIs of the respective applications was easy and the reverse mashup architecture relieved us from building user interfaces for these services. Both accessed services, Oryx and Twitter, offered JSON as data interchange format, while the components of our mashup, Xenodot on CouchDB and the process map in the Web browser, were essentially implemented using JavaScript. That, in turn, made data aggregation and service access surprisingly simple, yet effective.

Still, the lightweight integration of services that are available with no additional costs would not satisfy requirements for high reliability or performance, typically met by enterprise service buses. Time critical processes would suffer from the high latency of updating a process' state, which is in the range of tens of seconds to minutes, as well as the non-pervasive notification of participants. However, for most use cases in the context of collaboration management such requirements are not present. In our scenario of writing a scientific paper, this issue turned out to be negligible.

Twitter. Twitter proved very suitable to coordinate participants in an unobtrusive way. All participants were early technology adopters and use Twitter on a daily basis. Thus, Twitter fit well within our daily interactions. The reverse mashup architecture enabled every participant to use their Twitter client of choice on their desktop computer and mobile

---

[1]Twitter (`http://twitter.com`) has been designed as a public communication service. Therefore, users do not assume their conversations private.
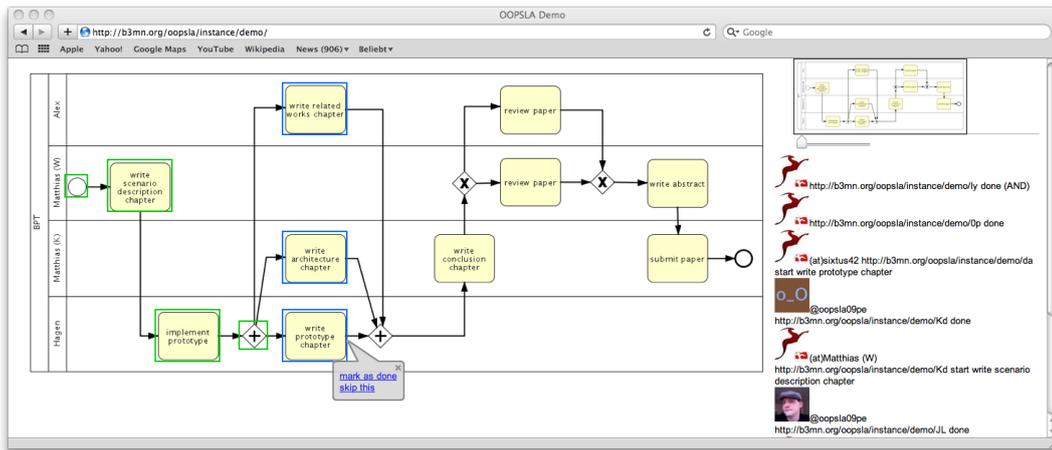
Figure 3: Process map, illustrating the state of a process instance.

phone. The restriction to few characters in Twitter messages forces authors to be short and focused, which supports quick comprehension of received messages on the one hand, but also fast responses on the other, thus reducing the amount of time spent to interact. Since we provide also a visually augmented map of the process, responding to commands is only few clicks away and blends seamlessly with the way we interact with the Web: discover through following hyperlinks.

The ability to comprehend the process in its entirety allowed us also to get in contact with other participants and to start discussions. Our expectations that such discussions need more than 140 characters and are carried out through face-to-face meetings, telephone and instant messaging rather than through Twitter messages proved to hold. However, using Twitter reduces the amount of emails sent to coordinate with each other; the process map gives immediate insight into the communication history, the process state, and upcoming tasks.

Process Flexibility. Above, we presented a simple process with few expressions in the modeling language and rather coarse-grained activities. It turned out that these means for simplification worked quite well. More meaningful expressions, such as events, could be used to include escalation scenarios, e.g., sending reminder messages, into the process model. However, we did not address process experts and, thus, kept the set of expressions small.

From time to time, we had to deviate from the ideal process to react to new situations, e.g., after a brainstorming session, it turned out that the related work section would be written by another author. Such changes have not been accounted for in the process. Since the process artifact—the paper written—was not strictly bound to the process coordination engine, this deviation did not cause significant problems. These deviations occurred due to the creative character of the process of writing a paper. Still, many less creative and rather static processes are conducted among people that can benefit from explicit coordination, e.g., processes of public administrations.

5. RELATED WORK

In this section, we focus on three areas of related research, namely process-driven collaboration management, service-based system integration, and mashups.

There is a large body of research on Computer Supported Cooperative Work (CSCW) and respective systems (cf., [27, 14, 37]). Good overviews of collaboration models and classifications can be found in [4, 13]. While collaboration management has been investigated from a variety of angles, we restrict our discussion to workflow driven collaboration approaches. Magdaleno et al. elaborate on the potential benefits of considering collaboration aspects in process modeling [26]. Based on an evaluation of the organization's collaboration maturity level, common descriptions of business processes are enriched with explicit collaboration activities. Such coordination, in turn, is at the core of process-aware collaboration systems. For instance, the Caramba system [12] aims at combining workflow, groupware, and project management functionalities to allow for holistic support of distributed collaborations. It, therefore, meets various of our requirements defined in Section 2.2. Still, the whole infrastructure is set up from scratch in a desktop application, whereas our approach is a mashup of existing functionality in order to be as unobtrusive as possible. Nevertheless, the functionality offered by Caramba might guide the choice of what to mashup in our setting.

Further on, our approach has to be related to the classical approaches for system integration. Starting with the SOA triumvirate of service provider, requester, and broker [7], service orientation became the predominant paradigm in system integration [2]. In particular, our approach can be seen as an enterprise service bus [8] lifted to the world of lightweight Web-based applications. Therefore, the research done in the field of enterprise application integration, e.g., on integration patterns [18] of which many can be found in existing mashup platforms, such as [41, 29, 36], provides the baseline for our human-centered integration scenarios.

In contrast to classic service-oriented architectures that focus on contractual integration of homogeneous interfaces (mainly WSDL/SOAP), mashups address a rather ad-hoc cooperation of heterogeneous resources [23]. Most of these resources provide information structured as collections of semi-structured documents, such as XML collections, database results, records scraped from Web pages. However, the diversity of mashed resources and mashup implementations, e.g., [21, 25, 28, 33, 38, 40], discloses the lack of common, standardized interfaces. This is a general weakness inhibiting the desired goal of making mashups universally easy to develop even for end-users [20], up to now. Although several attempts exist to homogenize interfaces and mashup-mechanisms, e.g., [1, 34, 38, 40, 42], these approaches are in early stages and compete with each other. Web-APIs providing functionality largely implement HTTP-based communication protocols, some complying with REST. However, due to the ad-hoc character of mashups, there are generally no contracts these protocols conform to, and thus require custom adapters at the mashup site. The reverse mashup architecture does not address particular mechanisms to ingest resources and Web APIs, cf., [22], and thus, is afflicted by the same issue.

6. CONCLUSION

In this paper, we first introduced the observation that much human interaction is performed in a rather structured manner and discussed that such implicit processes have significant drawbacks, i.e., they are often inefficiently conducted, not transparent to the participants, thus inconsistent, and not measurable. Process artifacts are often scattered over different places.

We proposed a lightweight collaboration management system that guides participants through collaboration efforts by making implicit processes explicit, that is, using process models to prescribe the interaction between participants. Using a message service that allows the correlation of publicly visible messages, we addressed most of the aforementioned deficiencies of implicit processes. Setting up such a system in the Web, we can refer to any artifact that has a URL.

Our prototype leverages typical mashup characteristics and properties, such as combining Web APIs to create new applications, yet reveals a new architectural paradigm for mashups: The reverse mashup architecture suggests using the services through their conventional application user interfaces and orchestrating the respective services in the background. For example, we use the Twitter API to coordinate collaboration in an unobtrusive way. A message is sent from the process execution engine, instructing participants to start their respective task.

In contrast to the classic architectural style of mashups that mainly addresses realtime insight among several resources at a certain point in time, the reverse mashup architecture is generally beneficial for mashups that need to provide a continuous service and, as such, suits the purpose to react to externally changed data and coordinate people in a collaboration process.

The presented prototype is an approach to understanding how collaboration can be supported in a social environment, applying knowledge from the domains of Business Process Management and Web 2.0. We believe that considerable benefits can yield from combining these disciplines and find this confirmed in similar scientific attempts elaborating on how business processes and novel social networking tools, such as Twitter, can be combined.[2]

7. REFERENCES

[1] S. Abiteboul, I. Manolescu, and S. Zoupanos. Optimax: efficient support for data-intensive mash-ups. *ICDE '08: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, Jan 2008.

[2] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services: Concepts, Architectures and Applications.* Springer, 2004.

[3] J. C. Anderson, N. Slater, and J. Lehnardt. *CouchDB - The Definitive Guide.* O'Reilly Media, Inc., 2009.

[4] G. Bafoutsou and G. Mentzas. Review and functional classification of collaborative systems. *International Journal of Information Management*, 22(4):281–305, August 2002.

[5] A. P. Barros, G. Decker, M. Dumas, and F. Weber. Correlation patterns in service-oriented architectures. In M. B. Dwyer and A. Lopes, editors, *FASE*, volume 4422 of *Lecture Notes in Computer Science*, pages 245–259. Springer, 2007.

[6] A. Bradley. Reference Architecture for Enterprise 'Mashups'. Technical report, Gartner Research, Sep 2007.

[7] S. Burbeck. Technical report, IBM Software Group, 2000.

[8] D. Chappell. *Enterprise Service Bus. Theory in Practice.* O'Reilly, 2004.

[9] L. Clarkin and J. Holmes. Enterprise Mashups. *The Architecture Journal*, 13:24–28, 2007.

[10] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. pages 137–150, December 2004.

[11] G. Decker, H. Overdick, and M. Weske. Oryx — sharing conceptual models on the web. In *ER '08: Proceedings of the 27th International Conference on Conceptual Modeling*, pages 536–537, Berlin, Heidelberg, 2008. Springer-Verlag.

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

[13] C. A. Ellis. A framework and mathematical model for collaboration technology. In W. Conen and G. Neumann, editors, *Coordination Technology for Collaborative Applications*, volume 1364 of *Lecture Notes in Computer Science*, pages 121–144. Springer, 1996.

[14] C. A. Ellis, S. J. Gibbs, and G. L. Rein. Groupware: Some issues and experiences. *Commun. ACM*, 34(1):39–58, 1991.

[15] G. Feuerlicht and W. Lamersdorf, editors. *Service-Oriented Computing - ICSOC 2008 Workshops, ICSOC 2008 International Workshops, Sydney,*

---

[2]`http://www.ariscommunity.com/users/mrosemann/`
`2009-08-07-where-bpm-and-Twitter-could-meet`

Australia, December 1st, 2008, Revised Selected Papers, volume 5472 of *Lecture Notes in Computer Science.* Springer, 2009.

[16] R. T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures.* PhD thesis, University of California, Irvine, 2000.

[17] S. Govardhan and G. Feuerlicht. Itinerary planner: A mashup case study. In E. D. Nitto and M. Ripeanu, editors, *ICSOC Workshops,* volume 4907 of *Lecture Notes in Computer Science,* pages 3–14. Springer, 2007.

[18] G. Hohpe and B. Woolf. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions.* Addison-Wesley, 2003.

[19] V. Hoyer, K. Stanoesvka-Slabeva, T. Janner, and C. Schroth. Enterprise Mashups: Design Principles towards the Long Tail of User Needs. In *SCC '08: Proceedings of the 2008 IEEE International Conference on Services Computing,* pages 601–602, Washington, DC, USA, 2008. IEEE Computer Society.

[20] T. Janner, R. Siebeck, C. Schroth, and V. Hoyer. Patterns for enterprise mashups in b2b collaborations to foster lightweight composition and end user development. *ICWS '09: IEEE International conference on Web Services 2009,* 0:976–983, 2009.

[21] W. Kongdenfha, B. Benatallah, J. Vayssière, R. Saint-Paul, and F. Casati. Rapid development of spreadsheet-based web mashups. *WWW '09: Proceedings of the 18th international conference on World wide web,* Apr 2009.

[22] M. Kunze. Business Process Mashups – An Analysis of Mashups and their Value Proposition for Business Process Management. Master's thesis, Hasso Plattner Institut an der Universität Potsdam, 2009.

[23] N. Laga, E. Bertin, and N. Crespi. A web based framework for rapid integration of enterprise applications. *ICPS '09: Proceedings of the 2009 international conference on Pervasive services,* Jul 2009.

[24] J. López, A. Pan, F. Ballas, and P. Montoto. Towards a Reference Architecture for Enterprise Mashups. In *Actas del Taller de Trabajo ZOCO'08/JISBD. Integración de Aplicaciones Web: XIII Jornadas de Ingeniería del Software y Bases de Datos. Gijón, 7 al 10 de Octubre de 2008,* pages 67–76, 2008.

[25] B. Lu, Z. Wu, Y. Ni, G. Xie, C. Zhou, and H. Chen. smash: semantic-based mashup navigation for data api network. *WWW '09: Proceedings of the 18th international conference on World wide web,* Apr 2009.

[26] A. M. Magdaleno, C. Cappelli, F. A. Baião, F. M. Santoro, and R. M. de Araujo. A practical experience in designing business processes to improve collaboration. In A. H. M. ter Hofstede, B. Benatallah, and H.-Y. Paik, editors, *Business Process Management Workshops,* volume 4928 of *Lecture Notes in Computer Science,* pages 156–168. Springer, 2007.

[27] R. Medina-Mora, T. Winograd, R. Flores, and F. Flores. The action workflow approach to workflow management technology. *Inf. Soc.,* 9(4), 1993.

[28] G. Miao, J. Tatemura, W.-P. Hsiung, A. Sawires, and L. Moser. Extracting data records from the web using tag path clustering. *WWW '09: Proceedings of the 18th international conference on World wide web,* Apr 2009.

[29] Microsoft. Microsoft Popfly. `http://en.wikipedia.org/wiki/Microsoft_Popfly` (discontinued since 09/2009).

[30] T. O'Reilly. What Is Web 2.0? Design Patterns and Business Models for the Next Generation of Software. `http://www.oreilly.de/artikel/web20.html`, Sep 2005.

[31] H. Overdick and M. A. Czuchra. PoEM - Potsdam Encoding for Models. *Services Computing, IEEE International Conference on,* 2:619–620, 2008.

[32] C. Pautasso and M. Frisoni. The mashup atelier. In Feuerlicht and Lamersdorf [15], pages 155–165.

[33] A. Riabov, E. Boillet, M. Feblowitz, Z. Liu, and A. Ranganathan. Wishful search: interactive composition of data mashups. *WWW '08: Proceeding of the 17th international conference on World Wide Web,* Apr 2008.

[34] M. Sabbouh, J. Higginson, S. Semy, and D. Gagne. Web mashup scripting language. *WWW '07: Proceedings of the 16th international conference on World Wide Web,* May 2007.

[35] C. Shirky. Situated Software. `http://www.shirky.com/writings/situated_software.html`, Mar 2004.

[36] D. Simmen, M. Altinel, V. Markl, S. Padmanabhan, and A. Singh. Damia: data mashups for intranet applications. *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data,* Jun 2008.

[37] L. A. Suchman. Do categories have politics? the language/action perspective reconsidered. In *ECSCW,* pages 1–, 1993.

[38] G. Wang, S. Yang, and Y. Han. Mashroom: end-user mashup programming using nested tables. *WWW '09: Proceedings of the 18th international conference on World wide web,* Apr 2009.

[39] M. Weske. *Business Process Management: Concepts, Languages, Architectures.* Springer, 2007.

[40] E. Wohlstadter, P. Li, and B. Cannon. Web service mashup middleware with partitioning of xml pipelines. *ICWS '09: IEEE International conference on Web Services 2009,* 0:91–98, 2009.

[41] Yahoo! Yahoo! Pipes. `http://pipes.yahoo.com/pipes/`.

[42] J. Yu, B. Benatallah, R. Saint-Paul, F. Casati, F. Daniel, and M. Matera. A framework for rapid integration of presentation components. *WWW '07: Proceedings of the 16th international conference on World Wide Web,* May 2007.

[43] S. Yu and C. J. Woodard. Innovation in the programmable web: Characterizing the mashup ecosystem. In Feuerlicht and Lamersdorf [15], pages 136–147.

[44] M. zur Muehlen and J. Recker. How Much Language Is Enough? Theoretical and Practical Use of the Business Process Modeling Notation. In Z. Bellahsene and M. Léonard, editors, *CAiSE,* volume 5074 of *Lecture Notes in Computer Science,* pages 465–479. Springer, 2008.