Jadex WfMS: Distributed Workflow Management for Private Clouds

Kai Jander, Winfried Lamersdorf
Distributed Systems and Information Systems
University of Hamburg
Hamburg, Germany
Email: jander, lamersd@informatik.uni-hamburg.de

***Abstract*—Structuring an organization around its business processes has many benefits for both the processes themselves as well as the workflow and business process management within an organization in general. However, there are many challenges that make such a transition from a classical hierarchic to a state-of-the-art process organization difficult. In particular, traditional departments often resist loss of autonomy within their organization and, thus, may prevent successful implementation of business process management techniques such as process-oriented workflow management systems. Therefore, this paper proposes a flexible workflow management system architecture consisting of multiple parts that can be replicated and distributed within an organization's private cloud network. Based on distributed components of a workflow management system, it supports both organizations with legacy organizational structures as well as those which require increased autonomy of their respective organizational units. As a result, this system allows, on the one hand side, to manage some organizational units independently and to regulate them in a distributed, process-driven way while, on the other, still allowing the overall organization to exploit many advantages of a centralized workflow management system. The resulting system is based on experiences of a DFG funded technology transfer project aiming at applying previous research results in autonomous business process management to practical needs and requirements of a real production system application.**

I. Introduction and Motivation

A key area of Business Process Management *(BPM)* which deals with organizational structure is *Business Process Orientation (BPO)* [1][2]. Business process orientation attempts to restructure an organization around its processes by taking a *process view* of the organization. For this, the organization is structured to meet the requirements of its business processes and their tasks. Individual units of the organization should be focused on performing those tasks with the process itself assuming full control and responsibility for all actions taken within the organization. This means that organizational units such as departments, work groups and structural hierarchies essentially become service providers for the organization's processes.

However, while *pure* business process orientation has many advantages once implemented, it has a number of disadvantages regarding its implementation. The core problem is centered around the fact that it shifts control from individual organizational units to the central authority of the process management, both with regards to the processes itself as well as the technical infrastructure if parts of the processes are supported by a workflow management system. This means that organizational units suffer a loss of both power and autonomy.

This has two important consequences which make pure business process orientation difficult. First, there is a social aspect: People within individual organizational units generally do not welcome loss of power and autonomy and will attempt to resist the change towards business process orientation. While this behavior may not be entirely rational, it is commonly experienced when attempting to reorganize businesses to align them with their processes [3]. The second aspect involves the concentration of power in the hands of the process engineers and their management. This can be detrimental in cases where organizational units are intentionally independent. This may be necessary in cases where one unit is supposed to regulate or control other units. An example would be an internal affairs unit of a police force with regards to other units of the police.

In order to resolve this problem, approaches that form a compromise between traditional separate organizational units and the service view of pure business process orientation. For example, one approach is matrix process orientation, where old departmental units are retained in a reorganized form and the processes are introduced as independent units of their own [3]. As a result, this preserves some of the autonomy of organizational units such as departments while still introducing a process view to the organization.

However, this approach undermines the traditional approach to automating business processes by implementing them as workflows. As a result, a lot of techniques have been used to support such hybrid organizations which attempt to bridge the two organizational principles of traditional departmental and hierarchical structures with process-oriented and service-centered approaches and even cross-organizational linking of business processes. However, useful techniques from workflow management such as handling and distributing work items from user tasks according to organizational models like roles, aspects of business activity monitoring and access control are often addressed on a localized and ad-hoc basis.

In fact, this often prevents the deployment of traditional workflow management systems, because they require a form of centralized control. On the other hand, deploying multiple heterogenous systems increases compatibility issues and increases the difficulty of monitoring or may even prevent monitoring

altogether. In order to address this issue, a single system is required which is flexible enough to allow organizational units some autonomy.

Another aspect is the increasing use of mobile systems, which have a number of special conditions and requirements. For example, access to the network may be limited or intermittent, which requires the ability to hold and process information locally. Mobile devices are also energy-constrained, depending on limited energy storage. Contrary to the previous requirement, this necessitates the use of networked resources to reduce energy consumption spend on processing information. As a result, a mobile device should use networked resources if available and seamlessly fall back to local resources if the network becomes unavailable.

This paper attempts to address these issues by proposing a distributed, self-organizing and flexible workflow management system based on Jadex [4]. The individual Jadex nodes can be deployed on hardware-based systems or This system consists of multiple components which can be deployed within a Jadex Cloud by different organizational units addressing their specific needs while still providing users of the overall system with a transparent view similar to a traditional workflow management system.

Jadex Cloud is a "Platform-as-a-Service" (PaaS) cloud system [5] which provides the basic infrastructure for distributed cloud applications. Like most PaaS platforms such as Paremus[1], Jadex Cloud allows the execution of custom cloud-based software without managing individual, potentially virtualized machines commonly found in "Infrastructure-as-a-Service" (IaaS) systems, nevertheless its deployment can be based upon an already existing IaaS infrastructure. By using Jadex Cloud as a base, Jadex WfMS can take advantage of some of the PaaS functionality provided by Jadex Cloud such as automatic node discovery and awareness, service infrastructure and communication and remote deployment and execution for workflow models. In addition, the use of dedicated systems for the workflow management system avoided due to the use of the common cloud resource pool. In return, the Jadex Cloud system gains additional workflow capabilities, thus improving the platform as a target for workflow-based applications.

II. System Requirements

The goal for the workflow management system presented here is to provide increased autonomy and robustness to organizational units regarding typical workflow management functionalities. Since each organizational unit can implement its sub-processes independently and can encapsulate them as services on the process level, distributed workflow enactment was considered less important. As a result, the focus of the system should be functionalities outside the enactment domain, such as work item distribution, monitoring and access control. In particular, each organizational unit should be able to augment the functionality of the system with regard to each specific aspect, allowing for partial control over aspects of workflow management outside the enactment. This implies that the system should be decomposed into multiple components, each of which provides a certain aspect of the workflow management system as a whole. Each organizational unit should be able to add components to the system which interact using service interfaces and service calls. The system must transparently self-organize itself as a single, distributed workflow management system and integrate all the individual components maintained by each of the organizational units.

As a first step towards an implementation of such a system, the cloud environment[4] was assumed to be friendly and cooperative, where, at least in principle, all the participants share similar goals and are willing to work together in a cooperative fashion. As a result, the target for such a workflow management system is a *private cloud*, which is a cloud system deployed and maintained within a single organization.

In order to achieve this goal of greater flexibility, the workflow management system and its components must meet certain requirements:

1) Each component must be able to be configured to the specifications of the organization.
2) The system must deal with component of the same type being deployed multiple times with different configurations.
3) The workflow management must provide its functionality as long as there is at least one instance of each functional component available.
4) The system should provide a certain amount of resilience against unexpected loss of components. In particular, even an exceptional termination ("crash") of a component should not impede the system as a whole.

These requirements imply a number of corollaries. First, requirement 3 precludes the use of static links between component services, each component must be able to switch to an alternative service if the previous one becomes unavailable. Second, in order to easily switch to alternative services, the use of a particular service component should be transparent to the service caller. Each service caller should be able to expect the same results without regard to the service component used. As a result, service components of the same type must coordinate with each other if a service call influences some common state. As a result, the system should be composed of a number of component types, each representing a functional aspect of the whole workflow management system. These components must be loosely coupled and self-organize automatically into a system that automatically adapts to changes to its infrastructure, providing a unified but dynamic and distributed workflow management system to the user.

III. Related Work

Workflow Management Systems are a well-established technology to manage human-centric and other tasks related to workflows, which represent the executable parts of business processes. The Workflow Management Coalition (WfMC) has established a reference model for the basic functionality of a complete workflow management system [6]. This model

[1] http://www.paremus.com/

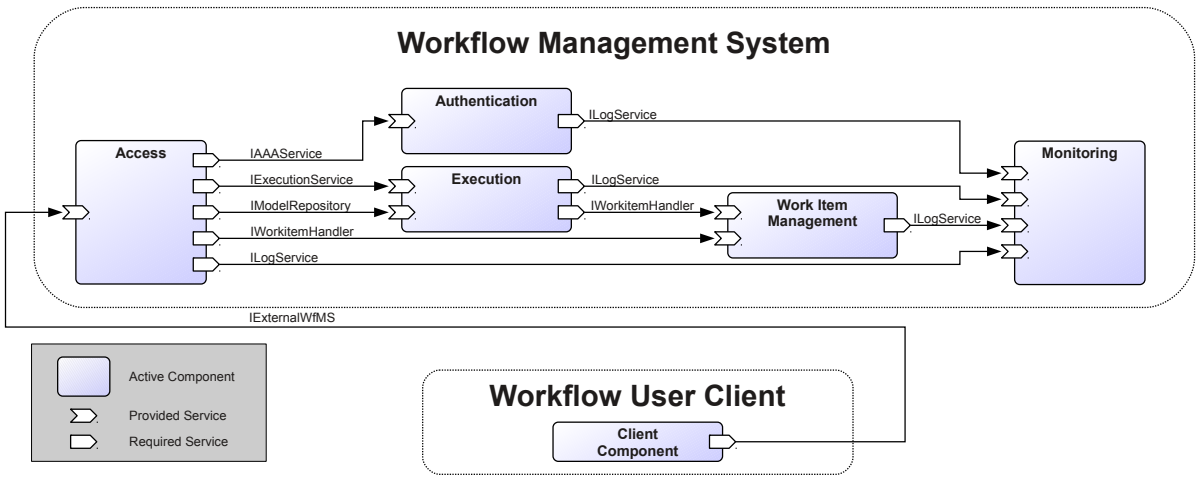

Figure 1. Architecture of the basic Workflow Management System without flexible distribution

already includes functionality for distributing the workflow enactment service part of the system and integrating them using a defined interoperability interface. However, unlike the approach presented in this paper, the enactment service is still a monolithic block and its functionality cannot be split and arbitrarily distributed. In addition, the approach to integrating multiple systems seems to be quite static and its fault tolerance concerning network and node failures appears to be fairly limited.

As a result, there have been multiple attempts to introduce greater flexibility for workflow management systems, most of which are focused on the enactment portion of the system. This includes distributed processes and distributed process execution. One approach attempts to fragment and distribute processes based on the *Business Process Execution Language* (BPEL) [7], enabling a distributed form of process orchestration [8] while still requiring a central coordinator. Another BPEL-based approach extends the original workflow model in order to achieve a distributed version of the workflow [9]. A SOA-based approach toward a more flexible workflow management system is presented in [10], however, the focus of the approach is mainly centered around workflow clients and does not address replication and resilience aspects.

Some approaches of distributed workflow enactment are based on multi-agent systems. In [11], the authors present an approach for a distributed workflow engine based on a number of autonomous software agents. The approach also integrates web services as part of the overall infrastructure.

A model for distributed workflow enactment based on petri nets is proposed in [12], which presents an approach based on tuple spaces. It primarily presents a model and infrastructure requirements for such a system, while useful extensions for the tuple space middleware are described in [13].

The focus of these systems seems primarily centered around the processes, the services they use and their enactment. Other aspects of workflow management like work items, monitoring, management of users and user accounts, access control and monitoring are rarely considered.

In contrast to these approaches, the approach presented here puts less emphasis on the distributed workflow execution and enactment. While the system allows distributed workflow execution using sub-processes, the focus of the system is primarily other aspects of workflow management like distributed work item management, access control, workflow model deployment and distributed monitoring. This allows distributed administration and management in multiple organizational units and support for partial recovery from certain failure conditions concerning network failures, node and component outtages and network splits to increase the reliability of the overall system and support deployment on special devices such as mobile systems.

The next section describes the general approach based on the requirements and goals for the system described in Section II. It presents the components used and the functionality they provide to the system as a whole. This is then used in the next section to describe the implementation of the system, followed by an example deployment, limitations and enhancements and the conclusions on the system.

IV. Approach

Jadex WfMS is based on an earlier and more traditional workflow management system for deployment in private cloud infrastructures [4]. This system was loosely based on the Workflow Management Coalition Reference Model but changed some aspects regarding the interfaces.

The system uses the concept of *Jadex Active Components* [14], which allows the direct execution of workflow models based on a dialect of the *Business Process Model and Notation* (BPMN) [15] and the execution of goal-oriented workflow models based on the *Goal-oriented Business Process Notation* (GPMN) [16].

The WfMC Reference Model describes five different interfaces, three of which can be used by human users to interact with the system. These interfaces derive their existance from the target object they address (workflow models, work items,

etc.). However, we found it more useful to group functionality in interfaces according to the users of the interface. For example, interfaces that provide functionality to a human user client are grouped into a single interface which unifies three interfaces in the Reference Model (Process Model Repository, Work List Handling and Execution) and adds an additional layer of access control. This approach was mainly chosen to simplify the interaction between client software and the Workflow Management System.

The basic architecture of this workflow management system can be seen in Fig. 1. In this architecture, the *monitoring component* receives, stores and provides events occuring within the system, such as initialization of new workflow instances, work item distribution and task completions. This information can be used for business activity monitoring purposes. The *work item management* is equivalent to the worklist handler in the workflow reference model and is responsible for distributing work items to users and receive work results. Workflow model management and execution is provided by the *execution component*, which provides both the model repository and workflow execution aspects of a workflow management system. This is due to the fact that workflows are executed as active components themselves and both their model and execution aspects are actually handled by the Jadex platform. As a result, the execution component wraps the platform services providing this aspect and supplies it in a more workflow-centric form: For example, it filters available component models for workflow models and provides a functionality for uploading and deploying new workflow models during runtime. The *authentication component* manages aspects regarding user clients and accounts. It authenticates connecting user clients and associates them with a user account, provides information about roles users have regarding workflows and roles they have regarding access rights to the system, such as the ability to start new workflow instances and monitor workflow execution. The access rights are then enforced by the access component, which uses the authentication component to authenticate incoming connections from user clients and verifies that the user account has the appropriate rights before forwarding a user client request to the component which is responsible for performing the request. For example, if a user requests to start a new workflow instance, the access component checks whether the user client is authenticated with a valid user account and then uses a second service call to ensure that the security roles of the user include the right to start new workflow instances. If both service calls to the authentication component succeed, the access component performs a third call to the execution component to start the new workflow instance. When the workflow instance has started, the user is informed about the successful execution.

Due to the use of active components, this system could already be distributed across multiple machines within a private cloud; however, it required exactly one of each component to be active, statically linked most required services and could not cope with addition or removal of components during runtime. As a result, there were few benefits to distributing the components aside from potential performance gains and most common deployments of this system remained on a single platform. Therefore, the components needed to be enhanced to allow more flexible deployment and self-organization in order to provide the required features. Another critical aspect of single platform deployment was that the users could circumvent the access component and its access controls by directly calling services on the other components of the workflow management system. Using multiple platforms lets the workflow management system rely on a feature of the Jadex Cloud middleware to prevent circumvention of the access component. This feature allows the segmentation of the cloud into separate logical *networks.* These networks share a common secret and prevent service call access from nodes unaware of the secret. This means that remote Jadex platforms only allow access to local services if the appropriate secret is known to the other platform.

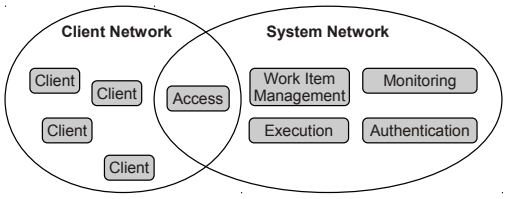

Figure 2. The Jadex Cloud is divided into two networks, the system network for the workflow management system proper and the client network for clients accessing the system

This allows the organization to divide the workflow management system into two separate domains (cf. Fig. 2). First, the system network contains all internal components of the workflow management system and second, the client network containing the user clients. The clients are unable to access the components of the workflow management system in the system network, however, the access component is available on a node which is available in both logical networks. This allows the users to call services on the access component which can then make service calls to the workflow management system components in the system network after applying the access controls, while being unable to directly perform service calls to internal workflow management system components. In order to support these functionalities, the five components of the workflow management system had to be modified to become autonomous and self-organizing. The next section provides an overview of the implementation of those services and how they interact with each other and self-organize to provide a transparent view of a single workflow management system to the user.

V. Implementation of the Components

This section deals with the implementation of required components to form a distributed, self-organizing and flexible workflow management system. The generally structure and division of functionality was based on the system described in Section IV and it still consists of five types of components. However, the components were enhanced based on the

requirements described in Section II so that any number of component instances can be deployed somewhere within the cloud. The system remains functional as long as there is at least one instance of each component available.

A. Authentication Component

The authentication component primarily provides three functions. First, it authenticates connecting clients using a list of valid user accounts. Second, it provides workflow roles associated with accounts which define which available work item from the process can be distributed to the user. Third, it provides security roles associated with the user account which define the access privileges of the user to the system: For example, a user may be allowed to request and process work items but may be prohibited from starting new process instances.

Since the system is supposed to support multiple instances of this component with different sets of user accounts, roles and security roles, a single authentication component must delegate work to other authentication components if local knowledge about a specific user account is unavailable.

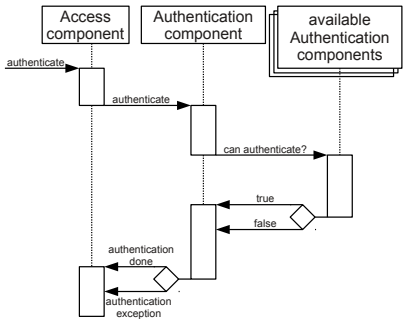

Figure 3. Interaction between the access component, the called authentication component and other authentication components during a client authentication

For example, Figure 3 shows the interaction of a client authentication. The access component receives a request for a user client to be authenticated. The access component then searches for the next available service interface offered by the authentication component. It then submits the authentication request to that authentication component. If the authentication component is not the one that holds the account information of the user of the client, it performs a service search for the service interface of other authentication components. It then successively performs a service call which asks the other authentication component whether the user is eligible to be authenticated.

As soon as an authentication component approves the authentication request, the *original* authentication component authenticates the client. If the list of available authentication services has been exhausted and none has approved the request, the request for authentication is denied. Note that if authentication is performed, it is done by the authentication component originally called by the access component, not the authentication component approving the request. The reason for this approach lies with the service search done by the access component, which attempts to find the local authentication component first. The authenticating component will also cache user account information such as roles and security roles, allowing users to continue working in the aforementioned failure scenario.

The implementation relies on the assumption that other authentication components are trustworthy. This is a limitation of the trust model of the system as it is currently implemented and one of the primary reasons the system is currently aimed at private clouds. Nevertheless, some protection is provided by carefully organizing access to the logical system network as discussed in Section IV. This prevents client nodes who do not possess the shared secret of the system network from using a malicious authentication component to inject invalid authorization approvals and access rights into the system. However, a malicious administrator of the system network could still provide such a component. As a result, the system network is critical for security. A potential approach for mitigating this problem is discussed in Section VII.

B. Execution Component

The execution component manages the deployed workflow models, allowing users to deploy new workflow models and remove old ones. In addition, it launches workflow instances if requested by a user or by another workflow instance. Since the workflow model deployment is based on platform support and is therefore always deployed locally relative to the execution component, a fairly simple delegation approach is sufficient to support multiple execution component instances.

For example, if an execution component receives a request to start a new process instance, it first checks whether the process model is available in its local repository, in which case the instance can be launched. If the process model is unavailable locally, it will start a service search for other available execution components. It then queries those components whether the model is available and once the right execution component is found, it will send an execution request to that component. Note that in contrast to the authentication component, the execution component holding the model launches the new process instance locally. This means that process instances are always launched on the node where the process model has been deployed. In case of failures such as network outages, the workflow models of the affected execution components become unavailable but other processes can continue so that the system remains functional.

C. Work Item Management Component

The work item management component handles work items issued by running workflow instances and offers them to eligible users based on their roles. In order to make distribution of multiple work item management components transparent to the user, the components periodically inform other work item management components about work items available to them using a service call. The work item is then offered at each of the work item components for processing by a user client.

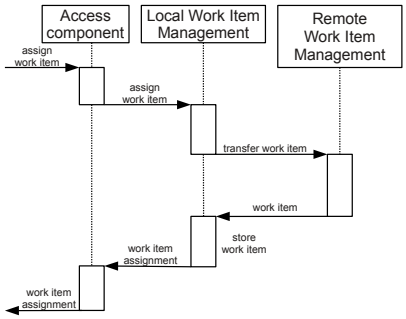

Figure 4. Transfer of a remote work item, access control interaction omitted for brevity

If a user requests a work item to be assigned to them, the access component attempts to find a local work item management component. If no local component is available, a remote component is used. The work item management component then checks if it manages the work item itself in which case the work item gets assigned immediately. However, if the work item is currently managed by a remote work item management component, it calls that component with a request to transfer the component (cf. Fig. 4). The work item is then transferred to the original work item management component which stores it along with its other locally managed work items before assigning it to the user. The advantage of this approach rests with the failure scenario of an intermittent network interruption. If the original work item component was available in the local network while access to the remote component is lost due to loss of a network connection, the users can continue working on the work item assigned to them and perform normal work item operations while the remote component is unavailable. Work results are then transferred to the issuing process once the network connection is restored.

D. Monitoring Component

The monitoring component receives events from running workflow instances and workflow components concerning certain aspects of its operation that may be interesting for a process engineer to monitor. This includes events such as creation of new workflow instances, generated work items, assignment and completion of work items and task execution. These events allow a workflow engineer or system administrator to monitor the system for either technical issues or issues concerning the performance of processes. In order to allow multiple monitoring components, the components must be able to exchange events. Since the events already receive a time stamp at the source, it is not necessary for the exchange to provide full ACID (atomicity, consistency, isolation, durability) guarantees. Instead, the approach used here follows the softer BASE (basically available, soft state, eventual consistency) principle with an option to further weaken the consistency criteria if it is not required. However, since the events receive the time stamp from their local source, the event order is only guaranteed on the local level unless adequate clock synchronization is provided.

During its creation, a monitoring component first attempts to find an existing instance of this component type. If one is found, it registers with the component for continuous exchange of events using service calls. If it fails to find a partner component for event exchange or loses its last one, it continues working under the assumption it is the sole monitoring component. However, this assumption is not necessarily true. For example, if two monitoring components start at the same time, the components may be unable to find each other because both are still initializing. A similar situation may arise when a network split occurs, the monitoring components lose their only event exchange partners but it becomes available again if the network split is resolved later. When those situations occur, the monitoring component may falsely assume it is the sole component in the system. In order to detect those situations, monitoring components that operate without an exchange partner occasionally perform service searches to find new monitoring components using the service search feature of the Jadex Cloud infrastructure which may optimize them using bindings, caching and announcement as appropriate.

During network splits and other connection losses, events are not exchanged with other monitoring components and therefore lack a complete set of events that occurred in the system. Each monitoring component therefore maintains an event log with all events it has received. This allows them to quickly update their set once communication problems are resolved. However, an unrestricted event log can grow without bounds and may consume resources unnecessarily. Therefore, monitoring components can be configured to occasionally purge older events if the event log exceeds a certain size. In case of communication, this can result in events never being transmitted to all monitoring components, however, since measurements are primarily done to identify issues, it is most often used within a specific time frame, such as during simulation [17]. In addition, if constant monitoring is used, it is often for statistical purposes which can tolerate small measurement errors. Nevertheless, if consistent monitoring is required, the event log can be configured to be permanent.

E. Access Component

The access component allows user clients to access the workflow management system and enforces access controls with the help of the authentication component. Since the access component is merely a gateway that relies on the other components of the system and does not hold any state of its own, it can trivially be deployed in multiple instances.

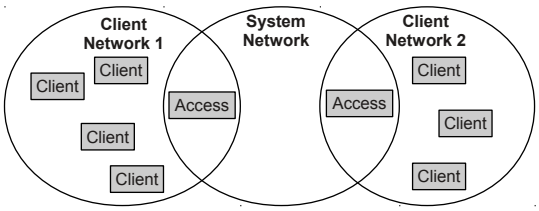

Figure 5. A workflow management system network with two access components and two different client networks

However, for the same reason there appears to be no reason for doing so. While providing multiple points of access can be done for performance reasons to allow parallel processing of user client requests, in most cases the processing done by the access component is fairly low since most of the work is delegated to other components.

Nonetheless, aside from potential performance gains, there is another reason why multiple access components may be desirable. As described in Section IV, the access components bridge two separate cloud networks, the system network which contains the components of the workflow management system and the client network, which contains the user clients that connect to the system. Since the networks are separated using a shared secret known to each node in the network, organizational units may each desire their own secret and client network in order to separately authorize nodes which can run user clients. For example, if a department hires a new employee, it wants to be able to set up the workstation for the new employee internally without having to request a central authority managing the shared secret to set up the user client. This can be done by establishing separate access components with the same secret for the system network and different secrets for the client networks (cf. Fig. 5). Each of the access components establishes its own client network by allowing access from clients with its client network secret while denying access from clients with another client network's secret.

VI. Example Use Case and Evaluation

A scenario which relies on the features of the workflow management system proposed here arose in the context of a production preparation process which was a focus of the Go4Flex project in cooperation with Daimler AG[2]. The production preparation process deals with preparing and validating a planned production process for a new car model [16].

Part of the process involves teams that validate the production process, both the construction parts of the vehicle and the process steps, by performing a test assembly of the vehicle. Issues that are identified during the process are later resolved in expert work groups. During the assembly, the teams use mobile devices to document the assembly itself and issues that may arise. However, network connectivity to the main network is often unavailable at the assembly location but becomes available once the devices return to the offices. This means that mobile teams are unable to communicate with the main workflow management components that later distribute documented issues to the relevant work groups. In addition, the expert groups tasked with resolving the issues found during the process are separated from each other.

In addition, since the process is based on exchange between experts in a number of fields, the process participants originate from a variety of departments such as part management and logistics. These departments often rely separate databases and other resources to address issues and suggestions which result from the process. As a result, the expert often use their own systems which need to be integrated in the WfMS.

[2]Parts of the process presented here have been simplified and abstracted for business secrecy reasons.

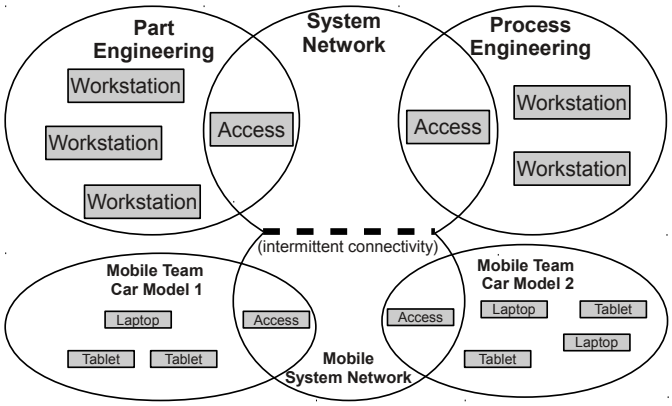

Figure 6. Deployment scenario for the production preparation process at Daimler

Figure 6 shows how this situation can be resolved using the proposed workflow management system. The mobile teams form a local network with a minimalistic replication of the workflow management system and form client networks for each of the car models. This lets them use the workflow management system while assembly is in progress. After the assembly the mobile devices are returned and the components of the mobile system network reconnect and synchronize with the main system network of the workflow management system. In addition, the expert groups have their own client networks for the workstations in their offices with each of them having a separate access component for connecting to the workflow management system. This allows them to manage their own workstations separately.

In the context of this deployment scenario the WfMS solves the particular challenges for the workflow developed for this scenario. A prototype implementation of the workflow and applications was tested by the process participants as part of a previously scheduled regular production preparation process on two separate occasion. The overall system performed adequately during those occasions and the process participants were generally satisfied with the prototype and it was transferred to the Daimler IT department for further evaluation and development.

VII. Limitations and Future Work

The current implementation of the system realizes all mentioned concepts and performs well. However, there are a number of general limitations and and some with regard to particular components. As with any distributed system, the workflow management system is ultimately constrained by the CAP-theorem [18]. This means the system in unable to guarantee consistency, availability and partition tolerance at the same time. For the prototype system, it was therefore necessary to make pragmatic choices regarding those guarantees. In most cases availability and partition tolerance was considered more important than perfect consistency.

For example, the monitoring component may not always have the complete set of events that occurred in the system due to connection loss. Consistency will eventually be restored if the event log is permanent, but not all events may be available at a given time. This is of particular importance if the event log is limited, which means that a situation may arise where consistency may never be reached since events can be lost which are never transmitted to the client. In other situations availability guarantees are abandoned, such as in the case of the work item management component. If the connection to the work item management component holding a particular work item is lost, that work item cannot be accessed until the connection issue is resolved. However, given the fact that in many cases multiple people can request assignment for a particular work item, access to a work item may also be lost because a different user gets the work item assignment. As a result, it was considered tolerable to allow restricted availability.

However, some implementation-specific limitations remain that could be resolved with some enhancements. In particular, the current trust model for the system is based on the assumption that the participants in the system are interested in cooperating. For example, each authentication component trusts other authentication components not to authorize malicious users or assign additional access rights to unauthorized user accounts. If the system is used within a single organization, this is an acceptable approach since it can be assumed that members of the same organization tend to share the goals of the organization. In addition, social remedies such as disciplinary actions are available to organizations to reduce malicious behavior. Nevertheless, an interesting goal for such a distributed system would be a cross-organizational deployment. In this case the above mentioned assumption no longer holds because organizations have different or even opposing goals, disciplinary actions across organizations are unavailable and legal remedies are not always an option. This means that a different trust model would be needed, where each component holds a public certificate of components it trusts and augments service calls with a signature. Currently the system does not support this approach. Nevertheless, depending on the trust relationship of the components, the approach may result in an inoperable system due to lack of trust or introduce unresolvable inconsistencies such as unassignable work items.

VIII. Summary

In this paper we proposed concepts for a flexible, distributed workflow management system with a special focus on problems beyond workflow enactment, specifically tackling issues like distributed work item management, event logging as well as user account and access management. The resulting system consists of multiple components providing a specific functionality, which can be replicated and distributed in order to allow for increased fault tolerance. In addition, the system allows organizational units to maintain some autonomy over their organization while still being able to participate in an organization-wide workflow managment system. This allows organizations to begin introducing a workflow management system gradually while transitioning into a pure process-oriented organizational structure or (partially) maintaining a more traditional organization structure in situations when process orientation is not an option due to external constraints.

While the proposed system certainly has some limitations, it is expected that some of these are eventually addressed. Furthermore, the system as it stands now works adequately for single internal deployments in organizations with the aforementioned constraints.

Acknowledgement: We would like to thank the DFG for supporting the technology transfer project Go4Flex.

References

[1] M. Weske, *Business Process Management Concepts, Languages, Architectures*. Springer Verlag, 2007.

[2] F. Leymann and D. Roller, *Production workflow concepts and techniques*. Prentice Hall PTR, 2000.

[3] W. S. H. J. Schmelzer, *Geschäftsprozessmanagement in der Praxis*. Hanser Fachbuchverlag, 2008.

[4] L. Braubach, A. Pokahr, and K. Jander, "Jadexcloud - an infrastructure for enterprise cloud applications," in *In Proceedings of Eighth German conference on Multi-Agent System TEchnologieS (MATES-2011)* (S. O. Franziska Klügl, ed.), pp. 3–15, Springer, 2011.

[5] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," tech. rep., July 2009.

[6] D. Hollingsworth, *Workflow Management System Reference Model*. Workflow Management Coalition, 1995.

[7] OASIS, *Web Services Business Process Execution Language (WSPBEL) Specification*, version 2.0 ed., 2007.

[8] L. Baresi, A. Maurino, and S. Modafferi, "Towards distributed bpel orchestrations," *Electronic Communications of the EASST*, vol. 3, 2006.

[9] R. Khalaf and F. Leymann, "E role-based decomposition of business processes using bpel," in *Proceedings of the IEEE International Conference on Web Services*, ICWS '06, (Washington, DC, USA), pp. 770–780, IEEE Computer Society, 2006.

[10] A. Hausotter, C. Kleiner, A. Koschel, D. Zhang, and H. Gehrken, "Always stay flexible! wfms-independent business process controlling in soa," in *Proceedings of the 2011 IEEE 15th International Enterprise Distributed Object Computing Conference Workshops*, EDOCW '11, (Washington, DC, USA), pp. 184–193, IEEE Computer Society, 2011.

[11] B. T. R. Savarimuthu, M. Purvis, M. Purvis, and S. Cranefield, "Integrating web services with agent based workflow management system (wfms)," in *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, WI '05, (Washington, DC, USA), pp. 471–474, IEEE Computer Society, 2005.

[12] D. Wutke, D. Martin, and F. Leymann, "Model and infrastructure for decentralized workflow enactment," in *Proceedings of the 2008 ACM symposium on Applied computing*, SAC '08, (New York, NY, USA), pp. 90–94, ACM, 2008.

[13] D. Martin, D. Wutke, and F. Leymann, "Tuplespace middleware for petri net-based workflow execution," *Int. J. Web Grid Serv.*, vol. 6, pp. 35–57, Mar. 2010.

[14] L. Braubach, A. Pokahr, and W. Lamersdorf, "Jadex active components: A unified execution infrastructure for agents and workflows," in *Intelligent Hybrid Medical Complex Systems*, Romanian Academy, 2012.

[15] Object Management Group (OMG), *Business Process Modeling Notation (BPMN) Specification*, version 1.1 ed., Feb. 2008.

[16] K. Jander, L. Braubach, A. Pokahr, W. Lamersdorf, and K.-J. Wack, "Goal-oriented processes with gpmn," *International Journal on Artificial Intelligence Tools (IJAIT)*, vol. 20, pp. 1021–1041, 12 2011.

[17] K. Jander, L. Braubach, A. Pokahr, and W. Lamersdorf, "Validation of agile workflows using simulation," in *Languages, Methodologies, and Development Tools for Multi-Agent Systems* (M. Dastani, A. E. F. Seghrouchni, J. Hübner, and J. Leite, eds.), pp. 39–55, Springer, 8 2011.

[18] S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services," *SIGACT News*, vol. 33, pp. 51–59, June 2002.