A Model and an Implementation Approach for Event-Driven Service Orientation

Olga Levina, Vladimir Stantchev
SOA and Public Services Research Group
Berlin Institute of Technology
10578 Berlin, Germany
Email: olga.levina|vladimir.stantchev@tu-berlin.de

***Abstract*—Event-driven architecture is gaining momentum in research and application areas as it promises enhanced responsiveness, flexibility and advanced integration. The combination of event-driven and service-oriented architectural paradigms and web service technologies provide a viable possibility to achieve these promises. This article is an extended version of an ICIW 2009 conference paper and introduces several aspects that can facilitate such combination. It presents an event model, outlines an architectural design and proposes sample implementation technologies. The ongoing evaluation in real-world scenarios confirms the applicability of the approach for the realization of web services-based event-driven architecture.**

***Keywords*-web services; event-driven architecture; service-oriented architecture; business events; business rules**

I. Introduction

Physical systems supporting business processes are increasingly coping with the effects of external changes and inputs. This information is used to monitor and control the process flow but it also creates new requirements for underlying network and application system structure. Asynchronous and data-centric communication in a distributed system is an approach followed by designers promoting event-driven and service-oriented architectures. Ubiquity and functional independence are some of the value adding characteristics of Service-Oriented Architecture (SOA). Asynchronous communication, interest-based message delivery using the publish/subscribe principle and event orientation by providing event sensors and event processing components are the characteristics of an Event-Driven Architecture (EDA). This article is an extended version of our ICIW 2009 conference paper [1] and motivates the implementation of a holistic architecture: Event-driven service-oriented architecture (ED-SOA) for combing function- and data-centric views on IT systems and enterprise as a whole. The combination of the two approaches is an actively discussed topic among information systems researchers, IT architects and vendors. This paper provides needed definitions and structures to promote common understandings and terms. Furthermore, reference architecture of an ED-SOA is proposed. Web services are suggested as the realization technology. This decision is confronted with the ongoing research and development results for enterprise event-driven systems.

The remainder of this article is organized as follows: in Section II we provide the definitions of EDA, SOA and web services. We introduce a reference architecture of an ED-SOA in Section III and present a realization approach based on web services and Quality-of-Service (QoS) assurance (Section IV). Related work on technology for the implementation of enterprise event-driven systems is provided in Section V. Discussion of our approach and outlook to future working areas complete the article.

II. Definitions

This section introduces some definitions that we use throughout the article.

A. Service-Oriented Architecture

Service-oriented architecture is one of the most discussed topics in the IT these days. Since there is no common SOA definition yet, the term is used as a combination of elements of software architecture and enterprise architecture. It is based on the interaction with autonomous and interoperable services that offer reusable business functionality via standardized interfaces. Services can exist on all layers of an application system (business process, presentation, business logic, data management). They may be composed of services from lower layers, wrap parts of legacy application systems or be implemented from scratch [2]. Service-orientation as a design paradigm roots in several already known approaches such as object-orientation, aspect-oriented programming (AOP), enterprise application integration (EAI) and business process management (BPM) [3]. Following service-orientation approach a system is decomposed in its functionalities. A service is hence an element that encapsulates a business function and cannot be further decomposed without harming its functionality. Services can be defined as autonomous, platform-independent entities that can be described, published, discovered and assembled [4]; they are technologically neutral, loosely coupled and support location transparency encapsulating business functionality [5]. There are different ways to implement distributed services into IT architecture. They can be implemented using data-based [6], object-oriented (e.g. CORBA and Java RMI) or service-oriented approaches. Since the data-oriented approach applies only to structured data [6] and object-oriented approaches do not necessarily enable loose coupling and

ubiquitous services access [7], service implementation today is often done using web services. Service orientation and SOA can be used best, when processes or their parts are standardized, when they are often repeated without changes, or when multiple users need the same process component to complete their tasks. Service invocation (consumption) in an SOA is realized remotely using RPC-like procedure and on request of the service consumer. This approach allows an explicit request for a WSDL-defined service interface to be invoked using SOAP message exchange.

B. Event-Driven Architecture

An event-driven architecture is a structure in which elements are triggered by events. An event in the enterprise context is a change in the state of one of the business process elements that influences the process outcome. Being abstract constructions, events are captured as event objects. An event object allows a machine to process, calculate and manipulate the event. Main components of an EDA are: event sources or generators, event recipients or consumers, event sensors and event processors. Event source(s) and event consumers are connected either directly (point-to-point) or via a middleware or broker (bus). Event source might be an application, business process, internal or external stakeholder or any other abstract data change [8]. Event recipients are all interested subscribers. Event capturing and delivery must be guaranteed by compatibility standards and can be processed in an extra component – the event agent. The logic of collecting and routing of events is captured in the event processor. Incoming event(s) are processed and forwarded to event consumers in (predefined and "soft") real-time. An event consumer reacts to received events by performing its functionality or publishing an alert. There are three types of events that need to be processed: single event, event stream and complex event(s). The difference between an event stream and a complex event can be described as event stream being a temporal sequence of event objects in the "first come-first-serve" manner [9] and complex events being a group of events that contains elements from different contexts or different time points. Processing events means performing operations on event objects like creating, transforming, reading or deleting. Algorithms for processing of multiple or interlaced events are summarized in complex event processing (CEP) technique. It allows identification and extraction of structured information from message-based systems. CEP includes event analysis and correlation delivering a decision triggering information. CEP uses business rules as well as patterns, maps and filters to specify relationship between events [10]. Event monitoring is facilitated by business activity monitoring (BAM) tools. These tools are often a part of a business process management suite and are currently more focused on detecting events and visualizing them on a dashboard than on automated decision making, therefore requiring less computational intelligence. Event-driven systems provide real-time visibility of the observed processes and allow almost real-time reaction.

In this article we show that a SOA can provide suitable conceptual structure for an EDA. Contrary to communication in SOA, EDA components interact asynchronously, event processor being a connector with high intelligence. In EDA event sources and event recipients do not know anything about each other, neither does event source know whether and what kind of reaction was caused by its appearance. Figure 1 shows an exemlary EDA architecture.

For further event processing and capturing during the requirements analysis or modeling phase, an event structure is needed. Figure 2 shows our proposed event model that allows a distinct description of any generic event. In this article we focus on business events, i.d. state changes of a business entity. This definition differs from the one in the context of event-driven distributed information systems like CORBA, where an event is defined as the occurrence of some interaction point between two computational objects in a system [11]. This kind of event or event description languages will not be considered for modeling, since state changes of business objects are our primary concern.

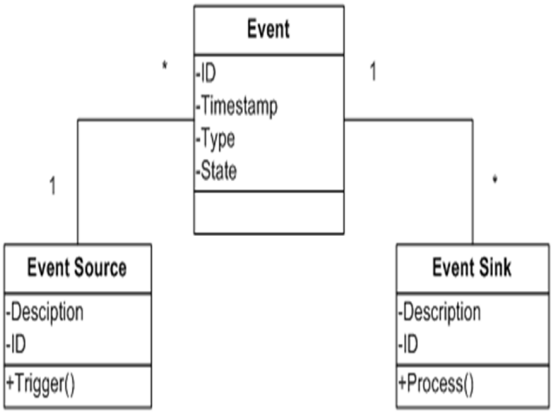

Figure 2. Structured Event Model

The event structure shown in Figure 2 shows the main components involved into event creation and processing. In the context of an event-driven architecture there are system elements that act according to the changes in states of other objects. That means that the event sources are being observed by the event sinks considering their change of states. The event source is described by its unique ID and a description, e.g., the name of the source, in natural language. Possible operation that can be performed by the source is triggering the event when the change of the state occurs. Here the main assumption is that an event, i.e. a state change, can originate only from one source. An event is identified by a unique ID, timestamp, event type and the current state of the event source. Timestamp is needed to compute the time

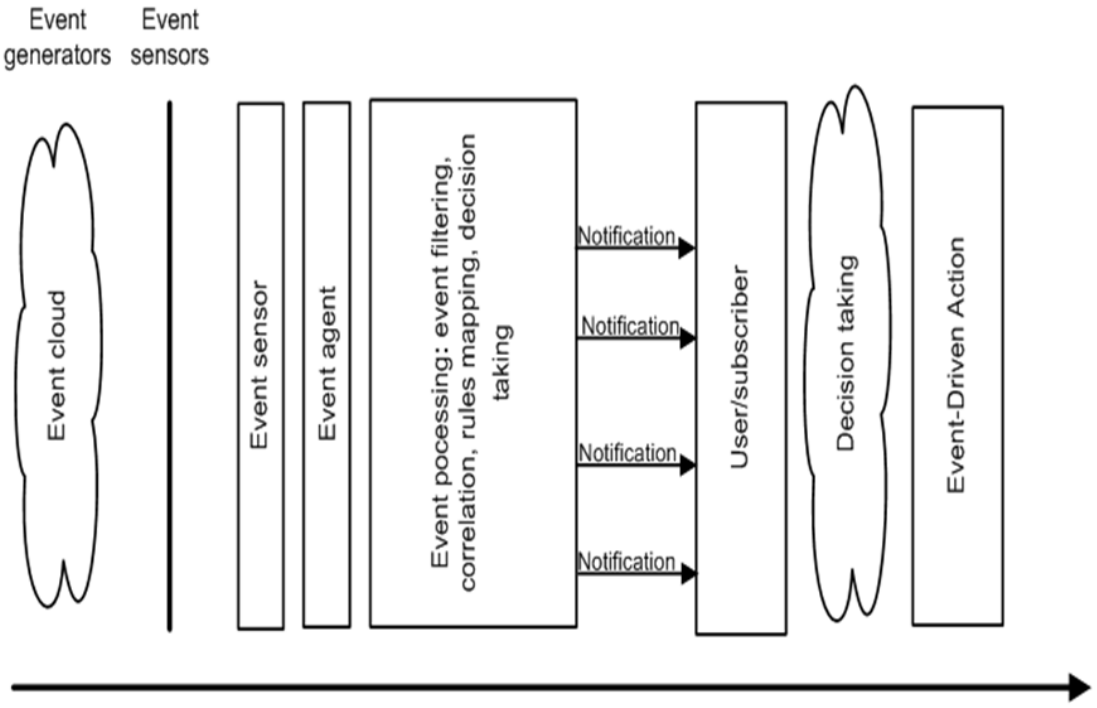

Figure 1. Event-Driven Architecture

model of the incoming events and to provide information for composition of complex events. Event type corresponds to the event type mentioned above: single event, complex event or event stream. This information can be processed by the sink in order to react to the incoming data. Event sink is an architectural object or element that is interested in the state change of the event source it is subscribed for, i.e. its actions are triggered by the state changes of the event source. Event sink description in natural language as well as an ID are used as its attributes. Event sink can also provide the first processing step of the incoming events. These processing can include queuing the events that are part of the event stream or combining the events to a complex event.

Our event model consists of the following formal elements:

- S is the set of event sources included in the model.
- S_n is the set of event sinks included in the model.
- Z is the set of the object states, while Z_s is the set of the object states Z of the source s.
- E is the set of events considered in the model.
- T is the set of possible event types, with: $T = single, complex, stream.$
- TS is the set of the timestamps, with: $TS = day, month, year, hour, minute, second.$

These aspects can be captured and modeled using a modeling eclipse Plug In, called Visual Event, Figure 3 shows the stand-alone event including event source, sink, and the event itself including its attributes and the data types of the attributes. It is also possible to comment on the model elements.

Using Visual Event Plug-in, it is possible to model all the events that are needed to trigger an action of the event sink using the annotation at the control flow. Additional information spaces are included in the diagram properties to take account of the sequence number of the event, timestamp, data type, etc. when modeling event sinks and sources. The Visual Event plug-in is comprised of an event, with event name, attributes and description, and an event trigger. The event trigger is the source that changes its states and thereby triggers an event. Event sinks are subscribers for a specific event occurrence in a publish/subscribe implementation paradigm.

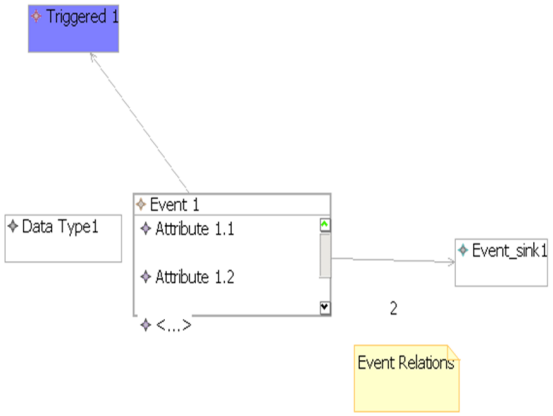

Figure 3. Event Modeling Plug-In

III. Combining EDA and SOA

Both SOA and EDA have characteristics that complement each other. Both use services but differ in the way they addressed a service to be invoked. SOA provides loosely coupled techniques like web services but its functionality is tightly coupled to the Request/Respond mechanism while EDA provides an asynchronous communication and loose coupling [12].

While SOA offers EDA a suitable design approach by providing a distributed environment for separating business logic, processes and technical functions, it benefits from another service invocation technique that loosens the rigor of the RPC-style calls. When observing these characteristics the merged structure of the three concepts provides, one can realize multiple synergy aspects. Service-orientation allows to capture and store events as services. Integration of legacy systems into service-oriented architecture may be done using the derived business rules the systems are using, or by using event-driven architecture. SOA is based on a remote access principle allowing a distributed environment, necessary for both event-driven architecture and business rules. EDA has a decoupled, asynchronous structure that complements loose coupling and synchronous communication of SOA [13]. Implementing SOA-suitable environment means implementing an environment where events can operate on their best and many architectural interactions are already standardized. Further synergies come up with communication and process management in a distributed system, which can be assured by adopting a business rules oriented ED-SOA. Often having a highly distributed architecture, enterprises create benefits from the real-time information availability. EDA provides a structure that allows a fast reorganization of business processes without affecting application or technical structures. Fast reaction to environmental changes in is possible without the need to adapt technical infrastructure. Functional decomposition on a high-granularity level, that is crucial for robustness to change of a system, is provided by SOA. Merging these concepts results in an enterprise architecture that is more flexible while being robust to changes. Its components are loosely coupled and can be accessed in any business situation.

The major aim of enterprise architecture is realized in the ED-SOA concept by SOA combining business functions and IT, and EDA focusing on data as well as business relevant event orientation; both SOA and EDA concepts can be used for application and legacy systems integration [14]. Covering the aim and component spectrum of enterprise architecture as described above, ED-SOA can be regarded as its evolution. Figure 4 shows a proposed ED-SOA reference architecture including security aspects, business rules processing and business data integration. Components that can be encapsulated as services are named. They were identified according to the main principles of service-orientation: their granularity is can be easily identified and discovered while being reusable by different components in different points of time. The concrete integration infrastructure into the application systems landscape depends on the technology used to realize ED-SOA. Here an enterprise service bus (ESB) is a suitable solution as the architecture is to be realized using web services.

IV. Enabling ED-SOA

After modeling the event and defining its specific structure, it can be realized technically using web service technology. Web services are currently the most promising service-oriented technology [15]. They use the Internet as the communication medium and open Internet-based standards, including the Simple Object Access Protocol (SOAP) for transmitting data, the Web Services Description Language (WSDL) for defining services, and the Business Process Execution Language for Web Services (BPEL4WS) for orchestrating services.

The Visual Event diagram (Figure 2)also delivers a XML-structure. Event content and its processing components such as sinks and sources can be derived from the event model as shown in figure 3 and implemented as a event service in a service-oriented architecture. This approach, first defining and modeling events for their further implementation using Web Services, allows a structured way to design and manage EDA conserving its main principle of agility and loose coupling. Modeling plug-in developed and presented here supports the easy implementation generating a XML-code of the event content. SOA provides important standards and tools, like WSDL and UDDI, for describing, storing and finding of the events within the architecture.

This section provides an overview of implementation technologies that we used in our proof-of-concept and is structured according to the elements presented in Figure 4.

Software components that call (consume) services can be developed in a variety of languages on a variety of platforms.

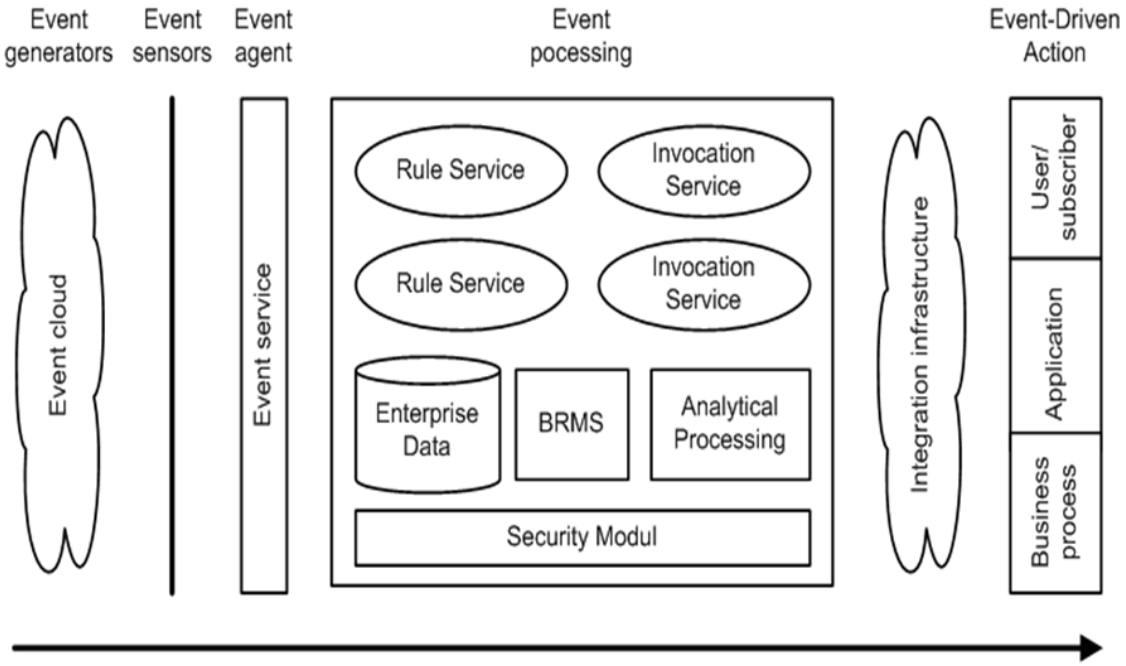

Figure 4. Event-Driven Service-Oriented Architecture

Typical integrated development environments (IDEs) allow this interaction without the need to code SOAP messages. They generate a proxy stub object on the local machine that marshals calls to the actual web service. Therefore, from a software engineering point of view a single service interaction is not much different from the interaction of COM (Component Object Model) or CORBA components. Important new aspect of web services is the promise of automatic composition going beyond the binary integration of COM and CORBA. Such flexible processing infrastructure can adapt more easily to changes in the functional requirements of an event-driven business process.

A. Platforms

The complexity involved in providing a single web service is often underestimated. A look at hardware platforms, even commodity hardware, reveals complex microprocessors and processing architecture. Standard OSs are far away from microkernel designs [16] such as Mach [17] and contain a large number of OS extensions. These are called *modules* in a Linux system [18] and *drivers* in a Windows system. [19]. Beside typical device drivers, extensions include network protocol implementations, file systems and virus detectors. Extensions are more than 70% of the Linux source code [20], while Windows XP includes over 35,000 drivers with over 120,000 versions [21]. Typical component frameworks such as .NET and J2EE often serve as the middleware for providing web services [22]. Therefore, we selected the .NET Framework as platform. A more detailed look at the application programming interfaces of these environments [23] and [24] reveals their complexity.

B. Quality of Service and Nonfunctional Properties

The nonfunctional properties (NFPs) of a software system are those properties that do not describe or influence the principal task / functionality of the software, but are expected and can be observed by end users in its runtime behavior [25].

QoS encompasses important NFPs such as performance metrics (for example, response time), security attributes, transactional integrity, reliability, scalability, and availability. Traditionally, QoS is a metric that quantifies the degree to which applications, systems, networks, and other IT infrastructure support availability of services at a required

performance level [4]. Web services environments are based on flexible composition of services and therefore demand greater availability of applications. Furthermore, they introduce increased complexity in terms of delivering, accessing and managing services.

The existing standards for specification of QoS characteristics in a service-oriented environment can be grouped according to their main focus: software design/process description (e.g. UML Profile for QoS and QML - QoS Modeling Language [26], service/component description (e.g. WS-Policy) and SLA-centric approaches (e.g. WSLA - Web Service Level Agreements [27], WSOL - Web Service Offerings Language [28], SLAng - Service Level Agreement definition language [29] and WS-Agreement [30].

Extensive research concerning NFPs also exists in the field of CORBA (Common Object Request Broker Architecture), particularly in the areas of real-time support [31], [32], replication as approach for dependability [33], [34], [35], [36], adaptivity and reflection [37], [38], as well as mobility [39], [40].

The approach we apply to formalize and control NFPs is called architectural translucency [41] – the ability to consider reconfiguration options at different system levels and understand their effects on the performance-related NFPs of a system. It allows us to specify service level objectives [42] and to enforce them by replication at different architectural levels, e.g., operating system [43] or service framework [44].

C. Implementing Rule and Decision Services

Our sample implementation uses the .NET Framework as a serviceware and the Microsoft Workflow Foundation (included in .NET 3.0) as basis for the rule and decision services. The workflow foundation supports different types of workflows (see Figure 5) and facilitates particularly the implementation of rules-based activities. Using it, we can map rules defined at the business level to any .NET programming language in a straightforward way.

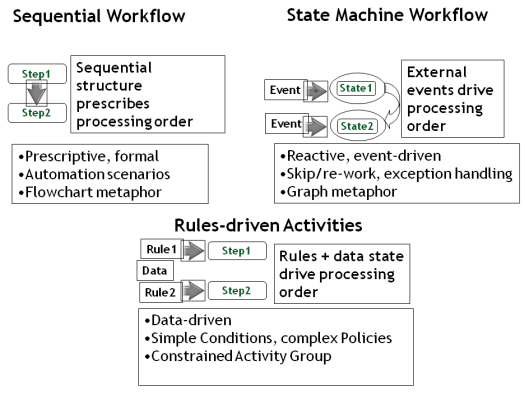

Figure 5. Support for Rules and Events in Microsoft Workflow Foundation (Source: Microsoft)

D. Implementing Invocation and Notification Services

Any step in our workflow (as implemented in the Microsoft Workflow Foundation) can call operations on other objects on the same machine, invoke other workflows or directly invoke web services. Events that trigger a state change (next step) of a workflow range from sensor information (e.g., RFID) through changes in data sources (e.g., relational databases) to web service outputs or fault messages. There are several integration depths that we regard as relevant for events:

- Events at the data level – here we differentiate between events originating from database management systems (DBMS), e.g., relational databases, and events originating from sensors, e.g., RFID readers and scanners.
- Events at the object level – these are typically state changes in class instances which we regard only if they are manifested by public methods.
- Events at the service level – a call, or a response of a service.

Generally, we can map the "lower level" events (data and object) to the service level using web service wrappers. Furthermore, we can combine events to complex events (e.g., a delivery has arrived and a warehouse is full) by using composite services. In the context of this composition we particularly regard the NFPs of the composed service, as described in [25].

E. Integration Aspects

An already agreed-upon SOA strategy greatly facilitates our approach as we can then expect that critical software functionality will be provided as web services in the specified timeframe. If our approach has to be integrated in more heterogeneous environments we can benefit from the capabilities of .NET 3.0 to interact with diverse remote components, such as other .NET objects, SQL servers and web technologies.

F. Application Scenarios

One application scenario that can greatly benefit from ED-SOA is logistics. Our demo application in this domain (more particularly contract logistics) differentiates between several states of a shipment that is being transported (see Figure 6). It begins with an initial event (*Container sent*) and goes through the following statuses: *Fetched, Accepted, Loaded, Unloaded, in Delivery,* and *Delivered*. Business users can define rules related to these statuses and corresponding events (e.g., a longer delay or a missed deadline) using a web-based user interface. We then use this rule specifications in our implementation to trigger next (or additional) steps in the workflow according to incoming events. Events can be propagated in a variety of ways: RFID-based communication in a warehouse system, e-mail notifications, changes in inventory databases, as well as other components or web service calls and responses. This makes our approach highly

flexible to changing business requirements – they can be submitted to our system as a new rule set via the user interface. Figure 7 shows an overview of our architecture on a given site (e.g., intermediate warehouse or point of delivery). It integrates an event processing component, components for event sensing, a supply-chain-management system, as well as to engines – the rules engine and the architectural translucency (AT) engine. The AT engine is responsible for service level enforcement with respect to NFPs.

One other specific domain that can greatly benefit from ED-SOA is healthcare. We have applied our approach in the area of clinical processes and their optimization based on localization techniques [45]. The requirements of the scenario – localization of a large number of mobile devices (10,000) within a refreshing interval of five seconds, make the architectural integration a challenging task. The scenario is described in details in [45], aspects of the service-oriented integration and service level assurance in [46].

G. System Evaluation

We conducted our evaluation twofold – using empirical evaluation methods as well as system-oriented performance evaluation. Our empirical evaluation follows the usablity evaluation methodology presented in [47]. In our healthcare scenario we used questionnaires and expert interviews as usability evaluation test methods. These were addressed at clinicians that use our system. An overview of the surveyed group is given in Table I, a summary of results is presented in Table II. Overall, there is a substantial ($\Delta > 50\%$) increase in usability.

To ensure QoS aspects in our application scenarios we apply the method of architectural translucency [41], [46]. In this article we present excerpts from the performance evaluation at one site of our logistics scenario. The specific replication configuration used is shown in Table III.

The results of our performance evaluation are shown in Table IV. We anonymized the names of the web services due to non-disclosure requirements.

Test results show that replication at the OS level improves performance by approx. 25%, while replication at the serviceware level leads to improvement by 5-7%. Dual replication led to improvement by 7-15%.

1) Confidence Intervals: Results in Table IV are average results from six consecutive test runs. Each test run included tests of every replication setting for 120 minutes with 1 second think time before a request. This corresponds to some 7200 requests that were sent to each setting.

All tests for Web Service 1 resulted in 7200 requests served for all replication settings, so here the confidence interval is clearly 100%. All other confidence intervals are between 99% and 100%.

V. Related Work

Distributed event processing and event-driven systems became popular in recent years as the technology needed to provide and support these systems is rapidly evolving. In the 1980s and 1990s message-oriented middleware was used to facilitate integration of various application systems within an enterprise. Basic event-processing can be regularized by inclusion of Java Message Service and message-driven beans in Java Enterprise Edition (J2EE) [48]. Message-oriented middleware allows a push-based, publish-subscribe data-centric communication through message brokers or queued messages. As for the embedded, real-time systems based on event-orientation, they are often written in languages such as C or C++, with real-time services provided by CORBA (Common Object Request Broker Architecture) [48], [31]. CORBA also provides a publish-subscribe mechanism by the CORBA/IIOP (Internet Inter-ORB Protocol) [49].

Composition of applications from web services is governed by different requirements than typical component-based software development and integration of binary components. Application developers and users do not have access to documentation, code or binary component. Instead, they rely only on a rudimentary functional description offered by WSDL. Services execute in different contexts and containers, they are often separated by firewalls and can be located practically everywhere. This leads to a set of specific requirements a composition mechanism must satisfy as identified in [50]: connectivity, NFPs, correctness, automatic composition and scalability.

Every composition approach must guarantee connectivity. With reliable connectivity, we can determine which services are composed and reason about the input and output messages. However, since web services are based on message passing, NFPs, such as timeliness, availability, and performance must also be addressed. Correctness of composition means that the NFPs of the composed service must be verified. Automatic composition is the ability to automatically perform goal-based composition. Finally, composition of services within SOA must scale with the growth of business services that are based on composed technical services.

With the native capabilities of web services fully developed, several approaches for service composition started to emerge. The first generation composition languages were Web Service Flow Language (WSFL), developed by IBM, and Web Services Choreography Interface (WSCI), developed by BEA Systems. However, these proposals were not compatible with each other, and this led to the development of second generation languages. The most popular of them is BPEL4WS [51], which is a joint effort of IBM, Microsoft, SAP, Siebel and BEA. It originates in the combination of first generation languages (WSFL and WSCI) with Microsoft's XLANG specification.

SWORD is an approach, together with a tool set, for rule-based service composition. Here a service is represented by a rule that expresses that given certain inputs, the service is capable of producing particular outputs [52]. A rule-

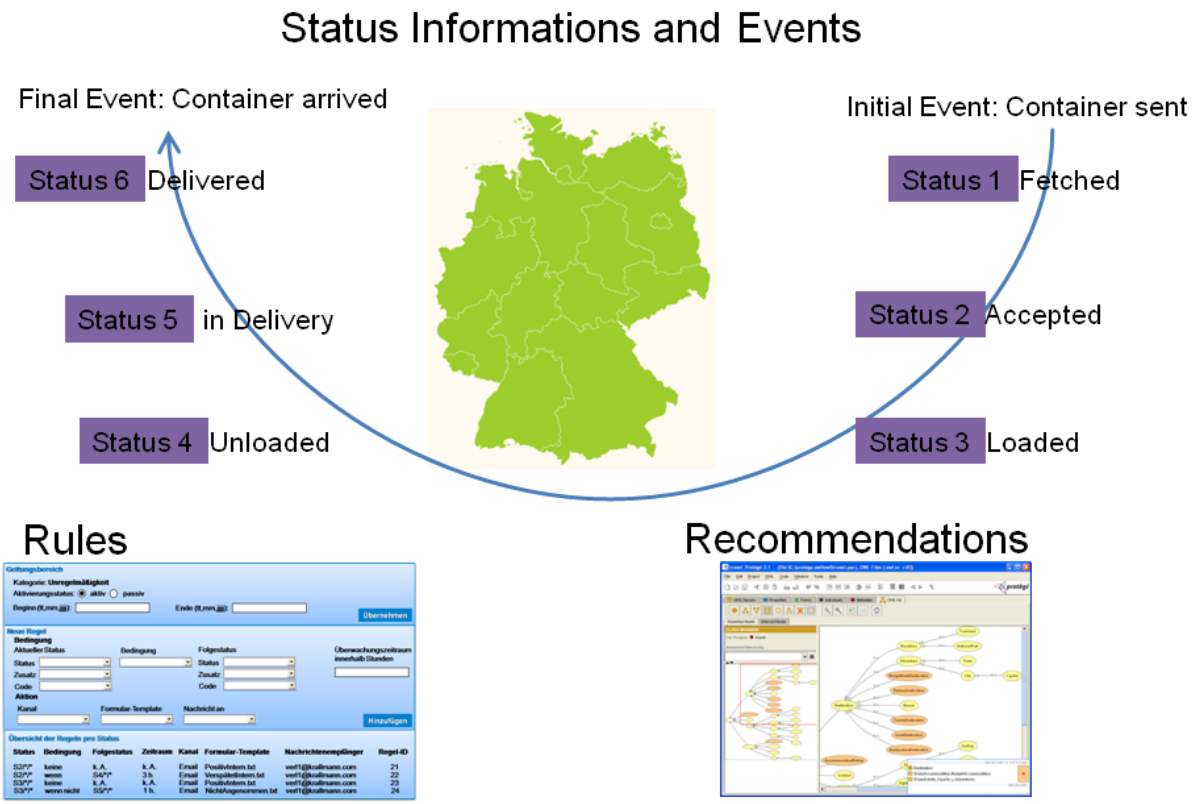

Figure 6. Application Scenario in Contract Logistics

Group Characteristics	No. of Participants	Percentage
Nurses	10	33.33
Surgeons	6	20
Anesthetists	3	10
Management	1	3.33
Other	10	33.33
Total	30	100

Table I
PROFILE OF TARGET GROUP FOR THE EMPIRICAL EVALUATION OF OUR HEALTHCARE SCENARIO

based expert system is then used to automatically determine whether a desired composite service can be realized using existing services. If so, this derivation is used to construct a plan that when executed instantiates the composite service.

Typically, SWORD does not require wider deployment of emerging service-description standards such as WSDL and SOAP.

Authors claim that although SWORD's expressive capabilities are weaker, the abstractions it exposes capture more appropriately the limited kinds of queries supported by typical web services which leads to simplicity and higher efficiency.

EFlow [53] is a platform for specification, composition and management of composite services. It uses a static method for workflow generation. Hereby a composite service is modeled by a graph that defines execution order of participating nodes. Graph creation is done manually, but subsequent graph updates can be automated. A graph may include service, decision and event nodes. Service nodes represent the invocation of atomic or composite services. Decision nodes specify workflow alternatives and decision rules. Event nodes allow services to send and receive certain types of events. Graph arcs show the execution dependency among nodes.

VI. DISCUSSION AND OUTLOOK

In this article we introduced the concept of an event-driven service-oriented architecture (ED-SOA) and proposed several aspects for its realization, such as an event model and a reference structure. Furthermore, we provided technology

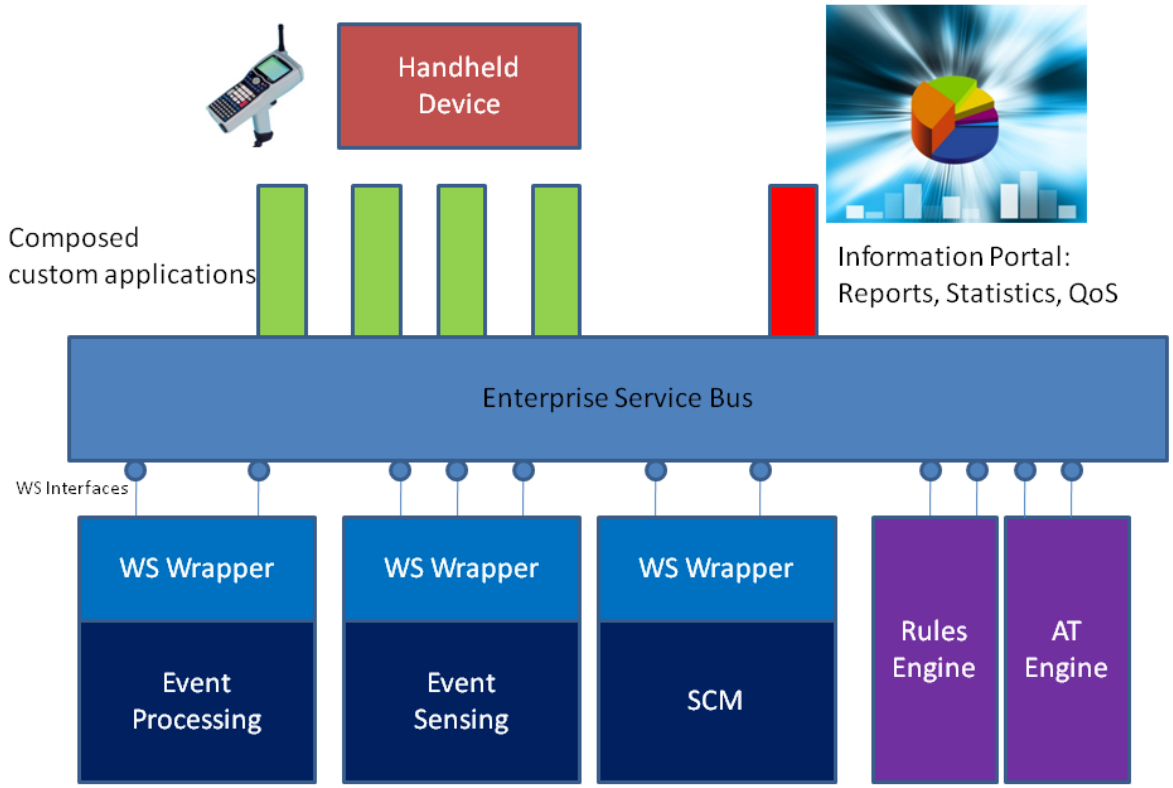

Figure 7. General Site Architecture for ED-SOA Implementation in Logistics: SCM denotes the Supply-Chain-Management System, AT Engine denotes the architectural translucency engine

Dependent Variables	**before**	**after**
Average Patient Preparation Time (min.)	31.20	16.30
Avg. Additional Preparation Tasks Needed (Nr.)	12	7
Avg. Number of Process Errors (perioperative)	6	2
Avg. Number of Process Errors (postoperative)	5	2
Clinicians: User Satisfaction (percent)	46	83
Patients: User Satisfaction (percent)	52	89

Table II
SUMMARY OF USABILITY EVALUATION RESULTS

and element definitions and outlined possible advantages of combining service-oriented and event-driven approaches for which we proposed a reference architecture. We regard our holistic approach as an important contribution that builds on many related concepts currently under development in this area.

The article also presented two application scenarios. Our application scenario in contract logistics used web services and the .NET Framework as enabling technologies and demonstrated major benefits of the approach. The empirical evaluation of our approach demonstrated increased user satisfaction, while its performance evaluation provided results that show its applicability for the assurance of QoS aspects within ED-SOA. Our future work lies in the areas of incorporation of predefined rule sets for specific domains (e.g., environmental conservation, privacy and security, healthcare applications) in the approach. This will allow us to provide a generic rule set that can be customized and extended according to the specific user requirements. The customization will be supported by a reference process for projects we are currently designing. We are also working on the further integration of various high-assurance techniques [41], [46] into the approach.

Setting:	Replication OS	Replication S
Setting 1: No Replication	n	n
Setting 2: Replication at OS	y	n
Setting 3: Replication at S	n	y
Setting 4: Dual Replication	y	y

Table III
REPLICATION SETTINGS FOR QOS ASSURANCE

Setting / Call	Web Service 1	Web Service 2	Web Service 3	Web Service 4
Setting 1:	7200	4176	4245	3578
Setting 2:	7200	5342	5418	4120
Setting 3:	7200	4424	4312	3692
Setting 4:	7200	4861	4803	3711

Table IV
TEST RESULTS AT DIFFERENT ARCHITECTURAL LEVELS - .NET AND WINDOWS ENVIRONMENT.

REFERENCES

[1] O. Levina and V. Stantchev, "Realizing Event-Driven SOA," in *ICIW '09: Proceedings of the 2009 Fourth International Conference on Internet and Web Applications and Services*. Los Alamitos, CA, USA: IEEE Computer Society, May 2009, pp. 37–42.

[2] H. Krallmann, C. Schröpfer, V. Stantchev, and P. Offermann, "Enabling autonomous self-optimization in service-oriented systems," in *Proceedings of The 8th International Workshop on Autonomous Systems - Self Organisation, Management and Control*. Berlin, New York: Springer, 10 2008, pp. 127–134.

[3] T. Erl, *Soa: principles of service design*. Upper Saddle River, NJ, USA: Prentice Hall Press, 2007.

[4] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-oriented computing: State of the art and research challenges," *Computer*, vol. 40, no. 11, pp. 38–45, Nov. 2007.

[5] M. P. Papazoglou, "Service-oriented computing: concepts, characteristics and directions," *Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on*, pp. 3–12, 2003.

[6] C. Batini, M. Lenzerini, and S. B. Navathe, "A comparative analysis of methodologies for database schema integration," *ACM Comput. Surv.*, vol. 18, no. 4, pp. 323–364, 1986.

[7] Y. Baghdadi, "A business model for deploying web services:a data-centric approach based on factual dependencies," *Information Systems and E-Business Management*, vol. 3, no. 2, pp. 151–173, 2005.

[8] H. Herbst, G. Knolmayer, T. Myrach, and M. Schlesinger, "The specification of business rules: A comparison of selected methodologies," in *Proceedings of the IFIP WG8.1 Working Conference on Methods and Associated Tools for the Information Systems Life Cycle*. New York, NY, USA: Elsevier Science Inc., 1994, pp. 29–46.

[9] B. Michelson, "Event-driven architecture overview- event-driven soa is just part of the eda story," Patricia Seybold Group, Tech. Rep., 2006.

[10] D. C. Luckham and B. Frasca, "Complex event processing in distributed systems," Stanford University, Tech. Rep., 1998.

[11] C. Ma and J. Bacon, "Cobea: a corba-based event architecture," in *COOTS'98: Proceedings of the 4th conference on USENIX Conference on Object-Oriented Technologies and Systems*. Berkeley, CA, USA: USENIX Association, 1998, pp. 9–9.

[12] J. van Hoof, "How eda extends soa and why it is important," 2.10.2008 2006.

[13] J. Pick, *Geo-Business: GIS in the Digital Organization*. Wiley, 2007.

[14] A. Kumar Harikumar, R. Lee, C.-C. Chiang, and H.-S. Yang, "An event driven architecture for application integration using web services," *Information Reuse and Integration, Conf, 2005. IRI -2005 IEEE International Conference on.*, pp. 542–547, Aug. 2005.

[15] S. Weerawarana, F. Curbera, F. Leymann, T. Storey, and D. Ferguson, *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More*. Prentice Hall PTR Upper Saddle River, NJ, USA, 2005.

[16] B. N. Bershad, C. Chambers, S. Eggers, C. Maeda, D. McNamee, P. Pardyak, S. Savage, and E. G. Sirer, "Spin - an extensible microkernel for application-specific operating system services," *SIGOPS Oper. Syst. Rev.*, vol. 29, no. 1, pp. 74–77, 1995.

[17] R. Rashid, D. Julin, D. Orr, R. Sanzi, R. Baron, A. Forin, D. Golub, and M. Jones, "Mach: a system software kernel," *COMPCON Spring '89. Thirty-Fourth IEEE Computer Society International Conference: Intellectual Leverage, Digest of Papers.*, pp. 176–178, 27 Feb-3 Mar 1989.

[18] D. Bovet and M. Cesati, *Understanding the Linux Kernel*. O'Reilly Media, Inc., 2005.

[19] D. Solomon and M. Russinovich, *Inside Microsoft Windows 2000*. Microsoft Press Redmond, WA, USA, 2000.

[20] A. Chou, J. Yang, B. Chelf, S. Hallem, and D. Engler, "An empirical study of operating systems errors," *SIGOPS Oper. Syst. Rev.*, vol. 35, no. 5, pp. 73–88, 2001.

[21] M. M. Swift, B. N. Bershad, and H. M. Levy, "Improving the reliability of commodity operating systems," *ACM Trans. Comput. Syst.*, vol. 23, no. 1, pp. 77–110, 2005.

[22] G. Miller, "The web services debate: .net vs. j2ee," *Commun. ACM*, vol. 46, no. 6, pp. 64–67, 2003.

[23] "Java 2 Platform, Enterprise Edition (J2EE)," SUN Microsystems, 4150 Network Circle, Santa Clara, CA 95054, Specification.

[24] *The .NET Framework*. Microsoft Corporation, 2004. [Online]. Available: \url{http://www.microsoft.com/net/}

[25] V. Stantchev, *Architectural Translucency*. Berlin, Germany: GITO Verlag, 2008.

[26] S. Frolund and J. Koistinen, "Quality of services specification in distributed object systems design," in *COOTS'98: Proceedings of the 4th conference on USENIX Conference on Object-Oriented Technologies and Systems (COOTS)*. Berkeley, CA, USA: USENIX Association, 1998. [Online]. Available: http://www.usenix.org/publications/library/proceedings/coots98/full_papers/frolund/frolund.pdf

[27] H. Ludwig, A. Keller, A. Dan, R. King, and R. Franck, "Web Service Level Agreement (WSLA) Language Specification," *IBM Corporation*, 2002.

[28] V. Tosic, K. Patel, and B. Pagurek, "WSOL-Web Service Offerings Language," *Web Services, E-Business, and the Semantic Web: CAiSE 2002 International Workshop, WES 2002, Toronto, Canada, May 27-28, 2002: Revised Papers*, 2002.

[29] D. Lamanna, J. Skene, and W. Emmerich, "SLAng: A Language for Defining Service Level Agreements," *Proc. of the 9th IEEE Workshop on Future Trends in Distributed Computing Systems-FTDCS*, pp. 100–106, 2003.

[30] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu, "Web Services Agreement Specification (WS-Agreement)," *Global Grid Forum GRAAP-WG, Draft, August*, 2004.

[31] A. Polze and L. Sha, "Composite Objects: Real-Time Programming with CORBA," in *Proceedings of 24th Euromicro Conference, Network Computing Workshop, Vol.II, pp.: 997-1004*. Vaesteras, Sweden: Humboldt University of Berlin, Aug. 1998.

[32] W. Feng, "Dynamic client-side scheduling in a real-time corba system." in *COMPSAC*. IEEE Computer Society, 1999, pp. 332–333.

[33] P. Felber, R. Guerraoui, and A. Schiper, "Replication of corba objects." in *Advances in Distributed Systems*, ser. Lecture Notes in Computer Science, S. Krakowiak and S. K. Shrivastava, Eds., vol. 1752. Springer, 1999, pp. 254–276.

[34] V. Marangozova and D. Hagimont, "An infrastructure for corba component replication." in *Component Deployment*, ser. Lecture Notes in Computer Science, J. M. Bishop, Ed., vol. 2370. Springer, 2002, pp. 222–232.

[35] M. Werner, "Replikation in CORE, Bericht an das Graduiertenkolleg "Kommunikationsbasierte Systeme"," Oct 1996.

[36] P. Felber and P. Narasimhan, "Reconciling replication and transactions for the end-to-end reliability of corba applications." in *CoopIS/DOA/ODBASE*, ser. Lecture Notes in Computer Science, R. Meersman and Z. Tari, Eds., vol. 2519. Springer, 2002, pp. 737–754.

[37] P.-C. David and T. Ledoux, "An infrastructure for adaptable middleware." in *CoopIS/DOA/ODBASE*, ser. Lecture Notes in Computer Science, R. Meersman and Z. Tari, Eds., vol. 2519. Springer, 2002, pp. 773–790.

[38] S. Gutierrez-Nolasco and N. Venkatasubramanian, "A reflective middleware framework for communication in dynamic environments." in *CoopIS/DOA/ODBASE*, ser. Lecture Notes in Computer Science, R. Meersman and Z. Tari, Eds., vol. 2519. Springer, 2002, pp. 791–808.

[39] G. Biegel, V. Cahill, and M. Haahr, "A dynamic proxy based architecture to support distributed java objects in a mobile environment." in *CoopIS/DOA/ODBASE*, ser. Lecture Notes in Computer Science, R. Meersman and Z. Tari, Eds., vol. 2519. Springer, 2002, pp. 809–826.

[40] S. Adwankar, "Mobile corba," in *DOA '01: Proceedings of the Third International Symposium on Distributed Objects and Applications*. Los Alamitos, CA, USA: IEEE Computer Society, 2001, p. 52.

[41] V. Stantchev and M. Malek, "Architectural Translucency in Service-oriented Architectures," *IEE Proceedings - Software*, vol. 153, no. 1, pp. 31–37, February 2006.

[42] V. Stantchev and C. Schröpfer, "Service level enforcement in web-services based systems," *International Journal on Web and Grid Services*, vol. 5, no. 2, pp. 1741–1106, 2009.

[43] V. Stantchev and M. Malek, "Addressing Web Service Performance by Replication at the Operating System Level," in *ICIW '08: Proceedings of the 2008 Third International Conference on Internet and Web Applications and Services*. Los Alamitos, CA, USA: IEEE Computer Society, June 2008, pp. 696–701.

[44] V. Stantchev, "Effects of Replication on Web Service Performance in WebSphere," International Computer Science Institute, Berkeley, California 94704, USA, ICSI Tech Report 2008-03, February 2008.

[45] V. Stantchev, T. D. Hoang, T. Schulz, and I. Ratchinski, "Optimizing clinical processes with position-sensing," *IT Professional*, vol. 10, no. 2, pp. 31–37, 2008.

[46] V. Stantchev and M. Malek, "Translucent replication for service level assurance," in *High Assurance Services Computing*. Berlin, New York: Springer, 06 2009, pp. 1–18.

[47] V. Stantchev, "Enhancing health care services with mixed reality systems," in *The Engineering of Mixed Reality Systems*. Berlin, New York: Springer, 09 2009.

[48] R. Berry, P. McKenney, and F. Parr, "Responsive systems: An introduction," *IBM Systems Journal*, vol. 47, no. 2, pp. 197–205, 2008.

[49] D. Bauer, L. Garce´s-Erice, S. Rooney, and P. Scotton, "Toward scalable real-time messaging," *IBM Systems Journal*, vol. 47, no. 2, pp. 237–251, 2008.

[50] N. Milanovic and M. Malek, "Current solutions for web service composition," *IEEE Internet Computing*, vol. 8, no. 6, pp. 51–59, 2004.

[51] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, *et al.*, "Business Process Execution Language for Web Services (BPEL4WS) 1.1," *Online: http://www-106. ibm. com/developerworks/webservices/library/ws-bpel, May*, vol. 139, p. 140, 2003.

[52] S. R. Ponnekanti and A. Fox, "Sword: A developer toolkit for web service composition," 2002. [Online]. Available: http://www.citebase.org/abstract?id=oai:wwwconf.ecs.soton.ac.uk:226

[53] F. Casati, S. Ilnicki, L.-J. Jin, V. Krishnamoorthy, and M.-C. Shan, "eflow: a platform for developing and managing composite e-services," *Research Challenges, 2000. Proceedings. Academia/Industry Working Conference on*, pp. 341–348, 2000.

[54] R. Meersman and Z. Tari, Eds., *On the Move to Meaningful Internet Systems, 2002 - DOA/CoopIS/ODBASE 2002 Confederated International Conferences DOA, CoopIS and ODBASE 2002 Irvine, California, USA, October 30 - November 1, 2002, Proceedings*, ser. Lecture Notes in Computer Science, vol. 2519. Springer, 2002.