2014 IEEE 18th International Enterprise Distributed Object Computing Conference Workshops and Demonstrations

Flexibility in SOA Operations: The Need for a Central Service Component

Liane Will
Active Global Support
SAP AG/ University of Magdeburg
Berlin, Germany
liane.will@sap.com

Veit Köppen, Gunter Saake
Department of Technical and Business Information Systems
Otto-von-Guericke-University
Magdeburg, Germany
{veit.koeppen; gunter.saake}@ovgu.de

***Abstract*— Efficient, controlled, and managed IT operations are strongly influenced by the architecture of a solution. A service-oriented architecture has a high flexibility and can respond to environmental changes and requirements by adapting the service orchestration. To take advantage of this flexibility in SOA, the traditional operations system oriented approach has to be changed into a process oriented approach. Furthermore, this results in fundamental new requirements for operation tools and services. In addition to common central elements of SOA, another fundamental element is required for process-oriented operations: a central operations cockpit. We present which requirements are raised for the basic components of SOA and how the central operations cockpit is designed in SOA. We use as an example a subset of ITIL processes for operation.**

Keywords—SOA, operations; ITIL, monitoring, change management styling, administration tools, flexibility

I. INTRODUCTION

As the evolution to service-oriented architectures (SOA) started several years ago, the flexibility of SOA on the basis of exchanging, extending, and reusing services was emphasized. Software vendor focus on this advantage and develop their products accordingly, see for example [1] or [2]. In fact, advantages are raised especially in the business solution design. Flexibility, in this new development, changes requirements in operations of such SOA-based solutions. In this paper, we focus on flexibility with respect to service exchange. There is a big difference between client-server-based solutions and flexible SOA-based solutions. Client-server based solutions are more or less fixed in their technical component structure. They follow a system-oriented operation. In contrast, operations of SOA require a service-oriented view. Due to the fact that services are exchanged if required by business, you need to be able to adapt your operations independently on the used service technology. In the following, we show that the service orientation requires a new type of service for operations to take advantage of the flexibility capacity of SOA. This service has to be a central fundamental element in any SOA. As a conclusion, the definition and general understanding of elements in a SOA has to be extended. For practical adaptation, services as well as the common elements, like service bus, service repository, and services need to support this Central Operations Cockpit (COCo).

In the next chapter, we introduce fundamental terms and some basics. Chapter III describes content and processes of the so called solution operations. This is done with help of the IT Infrastructure Library (ITIL), introduced by [3]. In Chapter IV, we demonstrate SOA operation requirements with respect to the ITIL processes with an example related to incident and problem management. Furthermore, we identify new challenges in operations for flexibility of SOA-based systems in context of service exchange and loose-coupling of services. As the main contribution within this work, we summarize these new requirements on SOA-specific services with the help of our example in Chapter V. Additionally, we introduce the Central Operations Cockpit (COCo) and discuss its functionality in this chapter. In the last chapter, we conclude our work and give a brief outlook on future directions.

II. FUNDAMENTALS

In contrast to the introduction of client server architecture, the development of SOA is an evolutionary process. Accordingly, the definition of SOA also develops over time.

A. Basic components of SOA

There is a consensus that one key component is the service itself. The Organization for the Advancement of Structured Information Standards (OASIS) defines service as: "[…] a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description"[4]. Based on this, Melzer and Eberhard [5] define SOA as: „[…] a system architecture which allows the usage of diverse, different, and maybe incompatible methods or applications as reusable, and free accessible services and therefore, provides a platform, technology and language independent usage and recycling" (see also [6]). This definition is quite open and focuses on the properties of SOA and its services to determine SOA. Another approach for a SOA definition is given by Krafzig et al. [7], where it is defined in terms of its components: „a Service-Oriented Architecture (SOA) is a software architecture that is based on

978-1-4799-5467-4/14 $31.00 © 2014 IEEE
DOI 10.1109/EDOCW.2014.51

the key concepts of an application frontend, service, service repository, and service bus. A service consists of a contract, one or more interfaces, and an implementation". The authors define SOA with the help of its own components to describe the necessary properties. In contrast, Melzer and Eberhard [5] ignore the specification of the implementation. The technical realization is not described. However, there is a common understanding, e.g., [8] or [9], of the components described by Melzer and Eberhard and both definitions are reasonable without contradicting one another. We show that the number of SOA components needs to be extended by another component for the operations of a SOA-based solution.

To enable service exchanges, further information (or meta-information) on the involved services has to be made available. Therefore, the service repository is introduced to manage all available services in a landscape. This repository provides information about services, e.g., on interfaces, process logic, input and output, as well as technical information. The service bus can operate heterogeneously according to different interface technologies or communication methods. A unified graphical interface (GUI) is used as a central user access point to SOA-based solutions [7], [9]. Nevertheless, what a service needs to fulfill in detail, and which properties it has to possess, is not provided in the definitions. The OASIS defines service as "a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description."[4] Papazoglou and Heuvel [9] as well as Ertl [10] state a number of principles for developing services. However, a service could consist of one single, quite atomic activity or a sequence of activities. A process is built as a series of linked services which could also split and run in parallel dependent on intermediate outcomes. A process has a start state and an end state. The important criteria, which activities are combined to an orchestrated service are finally decided by the recycling and reusability approach described in the service definition. Beside functional properties related to a process, a service has non-functional properties which are for instance related to its operations. In this article, we focus on those non-functional properties.

B. Flexibility

The concept of flexibility in SOA systems is the opportunity to exchange and adapt services at runtime. A key requirement is full encapsulation of services and standardized interfaces. Nevertheless, there is not one common valid definition of the term flexibility. Evans summarizes different understandings into one unified matrix [11], which builds the basis for the further investigation in this paper.

SOA-based solutions can potentially excel in all dimensions of flexibility because of the opportunity to exchange services during runtime. Users take advantage because of the fast ability to adapt to new situations and the capability to reuse services [12]. In this paper, we analyze implications for operations of a SOA-based solution if you take advantage of the described flexibility on the basis of exchanging services on the fly.

III. Context of Operations

Which requirements are necessary come in general from IT service and support management as described in the IT Infrastructure Library (ITIL) [3], see also Fig. 1. ITIL was first developed by Central Computer and Telecommunications Agency (CCTA, UK) in 1989 and was last updated in 2007 and is now called ITIL III.

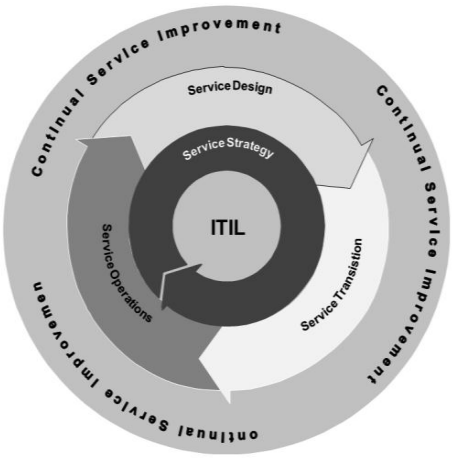

Fig. 1 ITIL Overview on services life cycle

ITIL processes and functions and their quality criteria are valid for all operations of SOA based solutions as well. ITIL describes processes, responsibilities and activities technology and provider independent. ITIL covers the tasks in IT service and support management along their lifecycle as shown in Fig. 1. The ITIL processes are assigned to the different phases in the life cycle of the services. In the phase of service operations the day to day activities are in focus. Nevertheless the fundamentals of the described processes and activities where developed in the earlier phases of the life cycle. By analyzing the processes step by step and checking their relevance for operations you get quite a complete picture of the high-level requirements of operations independent on the used technology. The current version of ITIL III is based on experience in operations of existing and former client server based solutions. The ITIL processes are affected in different ways by the architecture of the solution.

This article focuses on incident/problem management, change/release management and monitoring. The difference in operations between a client server based and a SOA-based solution is shown in the next section with help of the example of an occurred incident.

IV. System Operations versus Solution Operations

To demonstrate the differences in operations between client server and SOA based systems, we pick the typical situation of an incident raised by a user.

A. Incident/Problem Management

In client-server architectures there is almost a direct link between technical systems and activities necessary for

successful operations, e.g., a sales order process. For this reason, it is in most cases sufficient to operate each individual system according to technical indicators and therefore, it is common to call this system administration or system operations. If there is an incident within a business process or step the user can name the affected system including the error message, because she has logged on to the system and is working within that system. Additionally, error messages are managed within the system.

An essential advantage for the user of a SOA-based system is the decoupling of technical implementation and business functionality. There is no need for end users to interact with any of the technical components, systems, or hardware setup. The user sees the solution as a unit and is not interested or even aware of any technical components or services. From an operations perspective, this is often a disadvantage. In the case that the end user experiences an incident during execution of her business, she is not able to provide technical information on the root causing technical component or technical layer. In our example, the end user reports that there was a crash during the sales order creation. In client server based solutions, the root cause analysis is typically started in a technical component oriented way. This is an appropriate approach because the number of technical components in client-server architectures is restricted on a lower level. In most cases, the end user reports the name of the effected technical system or component. In SOA-based solutions this is very different. If you apply this method to SOA-based solutions, there are initial exclusions of technical components as root cause. In this case, there is a loss of the solution oriented view. With this approach, there is an increased risk of an incomplete analysis due to the restricted view within the system oriented approach. Accordingly, this requires a higher effort to localize the corresponding services and technical components responsible for the incident, and the incident has to be analyzed by many different experts, each doing her component oriented analysis and exclusion. It is also easily seen that this kind of indiscriminate search for a root cause requires an increased number of incident managers within the incident and problem management process. We state, it is necessary to perform an initial solution-oriented analysis step at the very beginning which requires a new type of tool to avoid an incomplete analysis in a SOA environment. In our example, the sales order process needs to be analyzed as a whole, independent of the service orchestration or the technology in place. As a first step the problem causing service needs to be identified. Neither the technical component nor the technical system is the unit to be analyzed. By investigating other processes directly or partly related to the operations of a SOA-based application we found similar issues. Will [13] gives an overview and discusses general challenges in operations in the case, you want to take advantage of SOA flexibility.

B. Service exchange

A very new challenge in SOA-based solutions is the opportunity to exchange services. As mentioned in client-server architectures, there is an almost fixed link between technical systems and activities. This is changing with the evolution to SOA. Services are loosely coupled during runtime [10]. So there is the opportunity to exchange or adapt services in a system. If you exchange a service in a complex SOA environment, certain prerequisites have to be taken into account. Two services can only be exchanged without changing the business process if they are identical in business logic, functionality, data, and interfaces. This requirement does not mean that the service needs to come with the same technology or operational requirements. However, input data for the service are not altered with a replacement of the old service but the new service has to create the same data format. The business logic has to be carefully considered in the new service and requirements on functional (and non-functional) properties have to be fulfilled. In order to substitute a service, the other service has to fit into the interface environment as well. It is a prerequisite that the output of the calling service can be transformed by the service bus into the required input for the new service. The output of the new service needs to be transformable by the service bus to create the required input for the follow-up service. Nevertheless, if there are services which fulfill all prerequisites related to the system architecture and an exchange from the functional perspective is possible, continuous operations must be ensured without any interruptions in addition. If a service exchange requires the adoption of operations, including adaption of operations knowledge related to the new service technology, a stable operation is not guaranteed. This can be solved by a unification of typical administration task for every service. In addition, this requires another new type of operations tool: a process oriented, standardized, central operations cockpit (COCo). Operating a SOA-based application with the same approach as in client-server-based system leads to:

- discounting the advantages of SOA,
- experiencing a rise in the cost of operations, and
- not fulfilling the common quality criteria of operations such as stability, availability, and performance.

Today, operation tasks are normally processed with the focus on technical components, systems, or databases. Each provider has its own

- system-specific tools,
- system-specific maintenance actions and quality criteria, as well as
- system-specific knowledge and experience required to use tools, execute actions, and understand quality criteria.

Examples of operating a SOA system are presented for instance in [14], [15], [16]. The tools partially cover services by other providers, but only if a suitable interface is available by the corresponding tool provider. However, there might be differences in the measurement units, meaning of figures handling and further more. So, specific knowledge and experience is necessary for each used service. In practice, this is in most cases not fulfilled by the operations team. As a conclusion, standardized operations with a tool independent on service specifics are required. This leads to additional requirements to services as well as the other components within a SOA. In the next section, we present those requirements and show how they can be fulfilled.

V. NEW CENTRAL OPERATIONS COCKPIT

The extensive usage of the flexibility of SOA-based applications is only an advantage if the exchange of services does not raise additional effort or instability in operations. As a conclusion, the operations need to be independent on the properties of the used services. Additionally, the solution operations have to be service- and process-oriented which requires a tool with just these properties. Such a tool can only be a central one, because this is necessary to fulfill the task of central monitoring and controlling, running or identifying services in the landscape. Therefore, we adapt the common understanding and definition of SOA components.

A. Extending the SOA Definition

Beside the common components of SOA, which are services (S), service repository (SR), and service bus (SB) [6], we introduce the new Central Operations Cockpit (COCo) as a required component in a SOA. We depict the SOA service lifecycle in Fig. 2.

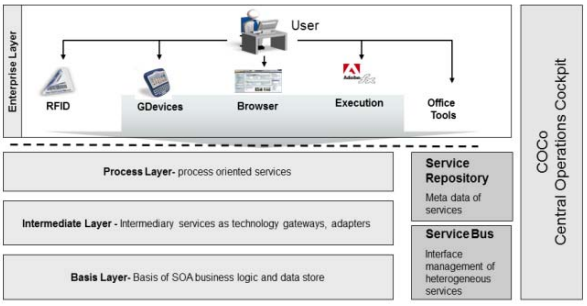

Fig. 2. ITIL Overview of services lifecycle

COCo is the integrating service within SOA which provides the central cockpit for execution of typical day by day operations. This service is technology independent and works in a standardized manner. Similar to both other central elements, it has to be supported by all services by providing necessary information and interfaces. SR and SB, which are also services, are reused by COCo, too. So, the service content has to be extended by supporting operations [4]. This requires a number of non-functional properties to support the IT service and support processes as shown in Fig. 3.

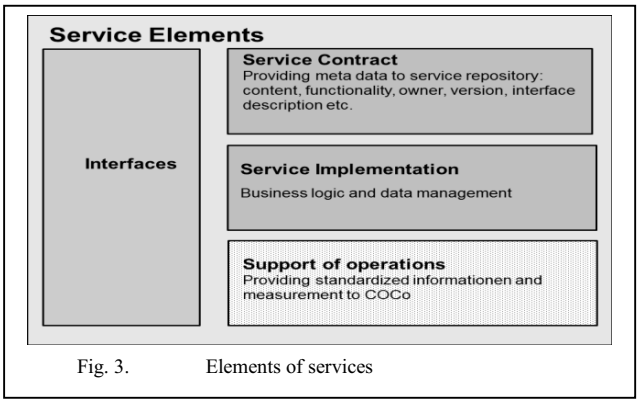

Fig. 3. Elements of services

ITIL has shown that it is possible to describe general accepted and standardized processes, roles, and activities in IT service and support management. Beside the mentioned differences in the operations of client-server and SOA-based systems, it is possible to provide such COCo.

In the following section, we describe properties of COCo and necessary extensions of already common elements of a SOA in more detail.

B. Extentsion of service repository

The service repository stores and manages meta-data of all services in the SOA. It is essential for the central availability of information about the used and available services in the SOA. However, Oasis defines UDDI [17] as a unified description language for the managed information by SR. However, which information is to be kept is not standardized. The initial challenge of the usage of flexibility is to evaluate which services are equal related to a system with the help of SR. [18] distinguishes between functional and technical (non-functional) services, which reduces the complexity related to the exchangeability of services related to the application landscape. Taking operations requirements into account, there is a need to adapt the SR. Nevertheless, Buchwald et al. [18] claim eight general requirements to SR, which we briefly reflect and extend with regards to operations. Note, our requirements SR are relevant for the service repository.

SR Requirement 1: Central management of service publication and contract

At every point in time, an operator can verify if a service is currently used in a solution or not. This requires that a service can be just used if it is published and the service contract is stored in the SR. A user agreement has to be available which clarifies the terms of usage. Furthermore, Papazoglou and Georgakopoulous [19] state, an outsourcer requires usage information to evaluate operations and license cost, too.

SR Requirement 2: Centralized Service Repository

It is important to separate domains in SR related to organizational units in a company. This follows the idea of more or less independent organizational units in a company which also requires independency in the SR. From the operational view, we cannot agree on this separation. Either the organizational units are independent, means they have different licenses, different usage contracts etc. which results in different service operations. Or they have at least partly aligned service usage rights. Then they are not independent from the operational view. Operations are always related to the service and cannot follow the domain idea. Independent organizational units also have independent business processes and therefore, different usage of the same services. If the organizations are not fully independent in the sense that they are using at least some services in the same way, we recommend a central SR. Independent domains require independent operations which mean copies of the service to ensure the different usage of the same service. This doubles the effort in operations of the same service. Therefore, from the operational point of view it is to be requested:

SR-Requirement 3: Information providing and usage to support operations

The authors in [18] describe the necessity to store functional as well as non-functional properties as, e.g., functional description of the services in SR. This does not cover information for the operations. We outline that this information shall include operational instructions and rules, too. What this means in detail, we describe in the following requirements.

SR-Requirement 4: Central information management for service life cycle management

Services follow different status in the life cycle management. Those are for instance design, build, operate, and optimize [3]. In addition, it is necessary that the status change of a service, e.g., change from development into productive usage, can be handled in a centralized way.

C. Change and Release Management

Change and release management requires a centralized management which shall not depend on the service technology. Change and release management need to be controlled across different life cycle statuses of services which includes the central change from one status to another in the life cycle. From the operational view, this information has to be available. It is not a must that it is stored in the SR, but it is reasonable to store it there.

SR Requirement 5: Management of relationships between repository objects

For exchanging services, there has to be an information source to evaluate which service calls which other service during runtime and which services are used at runtime within a specific process. System monitoring with focus on processes and services needs this information for example. SR is a suitable and reasonable store for this.

SR Requirement 6: Management assessment opportunities of service versions and planning

Buchwald et al. emphasize the importance of versions of services and planning [18]. New versions need to be planned. Old versions need to be retired. This requirement is closely related to SR Requirement 4. From our operational point of view, this requirement needs an extension. It is necessary that changes can be reported in relation to occurred incidents. This requires to store the time related dependency between changes in versions of configuration items as services and occurred incidents in the related processes.

SR Requirement 7: Long term archiving of process and service models to assess dependencies between incidents and changes

Buchwald et al. bring this requirement in relation to the traceability of business processes [18]. This traceability is also important for supporting incident and problem management. The stored information in the SR has to be evaluated in the case of an incident to check if there is a relation between a change and the corresponding incident.

SR Requirement 8: Administration of analyzable relations between functional and technical activities

Following Buchwald et al. [18], it is required that also the relations of functional and pure technical services during runtime related to an orchestration are stored. We emphasize this requirement from the technical view as well. It is a must to support especially incident and problem management with this traceability of processes. This information is also needed for the decision if a service is equal to another related to the solution and can be exchanged. The authors in [18] only investigate the SR-related requirements to decide either a service can be exchanged or not. The given requirements are valid, but not sufficient if we take into account how the service exchange affects operations. This leads to further requirements to SR and the other components for introducing the COCo. Note, we depict S requirements for the services and COCo requirements for our cockpit.

S Requirement 1: Complete encouragement to fulfill the requirements to SR

All given requirements to SR can be fulfilled only with help of the services which have to provide all necessary information to SR.

COCo Requirement 1: Complete encouragement and analyzability of the information available in the SR

Available information in the SR can only be used for operations if the data can be prepared and reported with focus on operations. Because SR is a central component in SOA, COCo has access to all information about used and available services and their relations as well as life cycle status. UDDI can operate as the general description language what simplifies the assessment of the data. The stored data in SR has to be independent on the technology of a service to avoid adoption issues if a service is exchanged. The same is valid for the presentation of the assessment of the data.

The change management is an important IT service management process to adapt an application to changing business needs. Several ITIL conform tools are available to support the change management process itself, e.g., CA Service Desk, HP Service Manager or ProactivaNET [20]. However, ITIL does not give any detailed rules on pure execution of the change. Every provider has product specific change execution tools depending on technology, e.g., [21], [22], or [23]. As already stated, the status exchange within the service lifecycle has to be managed centrally. ITIL assumes changes as directly related to a technical object. This can be a service in SOA as well. SOA introduces a new type of change: the exchange of an equal or similar service related to a business need. This requires suitable and valid information as basis to make a decision on the exchange possibility. As a result, there are the following main points from operational view:

- Central control and management of changes and releases;
- Central execution of movement of a service from, e.g., development to test environment;
- Solution oriented deployment which requires the synchronization of all depending changes in relation to a solution;
- Reporting across services and solution oriented reporting of changes related to a solution;
- Traceability of changes related to a solution.

A release is the bundling of a number of changes to a unit. Both processes are using the so-called Configuration Management Database (CMDB) defined by ITIL [3]. CMDB stores all configuration items properties and their relationship. A service is a valid configuration item. At the same time SR is the central element in SOA for storing all data about services. Therefore, the data exchange between SR and change and release management is essential. This leads to the following requirement for the SR.

SR Requirement 9: Integration of CMDB and SR to one central component

Change management requires development, possibly test environment, and productive environments. The movement of changed services from one environment to another has to be executed under the control of operations. As a conclusion, operations require information across the different environment.

SR Requirement 10: Central SR to integrate and control the different statuses of the services in the environments along the life cycle

SR Requirement 10 is an extension of SR Requirements 4 and 5. It is not sufficient to store the information on status only. Possible test or development environments need to be stored as virtual units similar to a production environment as shown in Fig. 4. As an example, we use a sales order process in Event-driven Process Chain (EPC) notation [24].

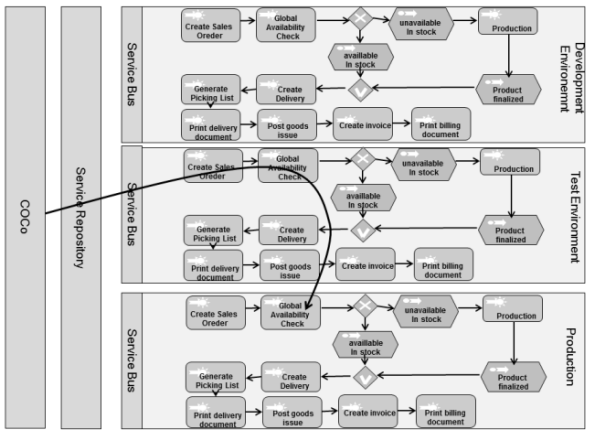

Fig. 4. Controlled transfer of a service "Global Availability" from test environment into production

The execution of the change from one environment into the other is done with help of the COCo in interaction with the SR. In Fig. 4, we assume that there are only one development, one test, and one production environment. Without general restrictions, there can be also more than three landscapes for different purposes. In practice, there might be a local SR per landscape. Nevertheless, there has to be one global SR. A hierarchical structure of local and one global SR is possible. For gaining overall control on statuses and versions of services the required meta-data of services have to be available and integrated at one central place, the global SR.

S Requirement 2: Standardized interface to change from one environment to another in the life cycle

As stated in the Section I, a universal, centralized COCo requires at the same time a partial standardization of services in the area of non-functional properties. If every service provides provider-specific technology-specific information and tools, this increases the effort in operations related to a service exchange. Services can be exchanged flexibly only if the effort and risk is as small as possible. From the service deployment point of view using the COCo, the encapsulation of technical properties is necessary. COCo initiates the status change via a standardized interface in the service. We assume to use the SB for encapsulation. Note, SB-requirements are related to requirements for the service bus.

SB Requirement 1: Standardized adapter for the service deployment interface

On the other hand this requires a standardized deployment interface at the COCo, too.

COCo Requirement 2: Standardized interface for initiation of status and environment changes of services (deployment interface)

All information about the deployment interface is to be stored in SR. This leads to:

SR Requirement 11: Storage for deployment interface information of services

S Requirement 3: Service responsibility for the information on its interfaces stored in SR

The information on the service interfaces has to be valid at any time. This is a prerequisite to provide input and output data in a suitable format. SR is the component to store all meta-data on services. Closely related to change and release management is the test management. Changes are verified before they can be released into production.

D. Test Management

Every change is either related to a single service or a bundle of services in a release. In practice, any change requires testing. Flexibility is related to fast adoption to new requirements and results in changes. Therefore, the execution of changes including testing needs to be fast and safe. The quality of tests is closely related to the quality of test environment. The test environment and the tested activities need to be comparable to the expected productive processes and data. This raises the problem, how to build a suitable test environment. Following the already presented requirements to SR, it stores all data about the services including different versions and interface descriptions. The SB needs to be able to prepare and transfer input and output data between services in the process flow. This has to be done automatically on the basis of interface information of a service in the SR.

SB Requirement 2: Preparation of input and output data in alignment with interface information in SR

SOA enables the service oriented testing due to the encapsulation of services. So, different levels of changes can be distinguished. In the case, a change affects only one service and does not impact its interfaces or relations to other services,

it is a capsuled change. For such a type of changes [25] proposes so called hooks as an extension of the SB. In Fig 5, we demonstrate how testing of a single service as *create delivery* can be executed with the help of hooks. Following the requirements SB1 to SR10 the service bus has all necessary information regarding the interfaces of services. With help of this information, a data flow comparable to production can be simulated. This results in a requirement for the SB.

SB Requirement 3: Simulation of data transfer from predecessor service and successor service aligned with interface descriptions for service stored in the SR

If this is a valid strategy for testing one single service it can be extended to a restricted number of services.

Independent on the architecture, it is a challenge to provide suitable test data in an environment. The simple copying of production data to test environments does not fulfill any security restrictions. This can be solved with help of already available services to copy and manipulate data, see for example [23, 24].

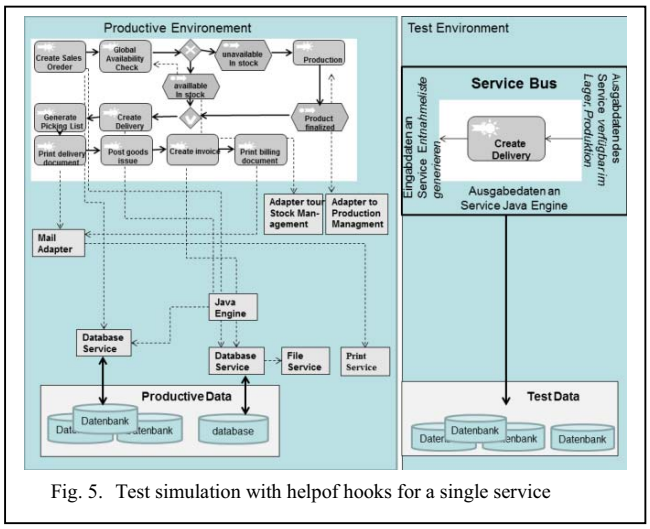

Fig. 5. Test simulation with helpof hooks for a single service

These are only some basic findings on testing which support and simplify testing. On this basis, it can be investigated how testing can be automated to reduce the test effort by multiple. However, we do not investigate this in the following. In the next section, we show the support of incident and problem management by the central components in SOA. Our approach solves the presented challenges related to the flexibility, see Section IV.

E. Incident and Problem Management

As outlined in Section VI.A, the new architecture of SOA and the usage of flexibility raises new challenges in the incident and problem management. The service orchestration of an application is not fixed. It can be adapted if necessary. This requires similar flexible tools related to the service exchange, but furthermore tools that are stable in handling. Stability relates to the similar usage and handling independent on the service and its technology. This can be only achieved with help of a standardized COCo. Instead of a system-oriented approach a service and process-oriented approach is required. Process-oriented means that, e.g., the root cause analysis starts with the complete process and its components, the services, in focus. Compared to existing tools and analysis strategies in client-server based systems, which are nevertheless system and technology-oriented, there is the need in SOA-based applications for tools as we describe in the following sections.

1) Process trace

In order to track the process flow, a service comprehensive trace is necessary. This includes the activity steps and corresponding meta-data. It has to be guaranteed: this trace or protocol is available for every service in a standardized manner. The standardization is necessary to avoid additional effort in orientation for operators in the case, a service is exchanged. However, such a trace occupies hardware resources, e.g., processing time or memory. Due to that such a trace needs to be central switchable with a defined scope, e.g., restricted to activities of a user or a single process. Such a process trace should contain:

- Executed services and activities in the service including input and output data,
- Start and end point in time,
- Occupied hardware resources, as CPU, input and output data amount.

Fig. 6 demonstrates the creation of a process trace reduced to the business step *create delivery* within the example sales order process. Each involved service protocols its important activity steps. All generated steps are transferred to the COCo which collects and orders in time. This results in the following requirements for services and the COCo.

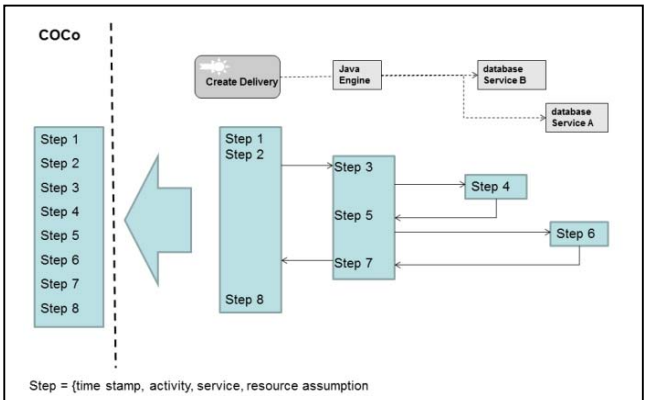

Fig. 6. Generation of process trace including

S Requirement 4: Provision of a switchable trace including service and activity information, time stamps, and resource consumption

Such a trace needs to be central switchable in the COCo for the defined focus, e.g., a process or a user including the automated cascade of switching the trace for involved services.

COCo Requirement 3: Central switching of process trace including all services

COCo uses the information provided in the SR about the service orchestration. It integrates the produced trace parts per service to a time ordered overall trace as shown in Fig. 6.

COCo Requirement 4: Provision of a standardized interface to services for the data transfer of trace pieces

COCo Requirement 5: Build an ordered summarized central process trace on the basis of the transferred traces for each service

The COCo integrates the generated trace data related to the execution of a service to a central aligned view. This enables operators to track the steps within a process or related to a user. In the case of an incident, the processes can be completely tracked. With help of time stamps, performance issues can be analyzed additionally.

2) Integrated Exception Analyzes

Another important task in day to day operations is exception handling. If a service used in a process creates an exception this can lead to a number of subsequent exceptions and finally, the whole process can be disrupted. As part of the problem analysis, it is important to identify the initial exception. This is to be done independent on the used technology of the services in a process. This requires traceable as well as understandable SOA wide information. This is only possible if every exception and its meta-data as the corresponding service, the relationships, and the time stamps are forwarded to the COCo. As a conclusion, there are further requirements to services and COCo.

S Requirement 5: Forwarding of standardized exception information

The standardization of forwarded information is necessary to ensure a general understanding of information independent of the used service and its technology. Typical information is raising service, time stamp, and exception content transferred data. COCo has to collect these data and provide a unified platform for monitoring and reporting. It has to reflect relations between exceptions and service order in a process.

COCo Requirement 6: Provision of a standardized interface for receiving exception data by services

COCo Requirement 7: Recording of the order of the raised exceptions and assignment to services and processes

Until now, a unified agreement on necessary information included in an exception is missing. Therefore, every developer decides on her own. In best practices, there are company rules for the content. In practice, information is technology dependent and developer specific. The extensive usage of service exchange requires a unification or standard for exception information.

S Requirement 6: Standardization of the provided exception information independent on service technology

The information provided by the services needs to be analyzed without deep and specific knowledge on the used technology. As soon as the analysis requires service specific know-how this is an obstacle for a potential service exchange. Exception monitoring and analysis takes place if there is already an incident. Therefore, it is reactive. Beside this reactive manner, monitoring can help to avoid incidents. A permanent measurement of typical quality criteria gives a prognosis for a critical development of quality criteria.

F. Monitoring and Service Level Management (SLM)

Another important task in the day to day business is the control of operations. In the context of event management, there have to be measured defined suitable quality criteria. A change in the status or reaching a threshold is a signal or alert. Typical monitored categories are availability, performance, throughput, exceptions, and security. The target is to measure these and only these quality criteria that directly or in combination signalize a critical development and require an activity to avoid a serious situation. Closely connected to monitoring is the service level management (SLM). However, the service level agreement (SLA) is a contract between an operations team and, in the widest sense, a user team of a system. The user team defines or provides the necessary quality criteria for their business, e.g., expected performance of a business process. The operations team needs to drill down these quality criteria to suitable technical and measurable criteria related to the technical solution. Accordingly, criteria are related to services and processes in SOA. Unfortunately, nowadays a SLA is often based on pure technical quality criteria which are neither related to the business processes nor to the services. As discussed in [28], the current technology of tools is based on data collectors developed for every specific service by a provider. The collected data are in addition itself service technology dependent and therefore, they require service specific knowledge for interpretation. Taking the advantage of the flexibility of SOA, based on service exchange, there is the need for monitoring independently from the service orchestration of a process. This requires the opposite approach: every service has to provide predefined monitoring data in a unified measurement unit to the COCo. As a first step, it is to standardize or decide which categories of quality criteria are necessary to monitor day to day. A first approach is delivered again by ITIL [3]. A service needs to provide which measured range of a quality criterion is uncritical and to give necessary thresholds. Furthermore, the service has to provide metrics and rules to decide on criticality.

S Requirement 7: A metric for the measuring key performance indicator

S Requirement 8: Provision of a rule set for the assessment of measurement results (metric of relationships)

S Requirement 9: Provision of measurement ranges for the assessment of measurement results

S Requirement 10: Measurement of quality criteria during its execution

S Requirement 11: Unified transfer of measurement data to the COCo

There are two general strategies to realize these requirements. Some of the quality criteria can be implemented via unified interfaces in the COCo. As an example, we give the processing time of a single service. Due to the communication handling by the SB, it is known to the SB when data are transferred to a service for processing and when the output data are ready. We conclude, the services have to communicate exclusively via the SB. The second opportunity is the strict standardization of services. Each service has to provide the necessary data in a unified format. This requires clear guidance

already during the design phase of services to provide the non-functional requirements. Nevertheless, the COCo needs to process the incoming data which means assess and report. If there are related quality criteria, the assessment becomes complex and needs a rules set provided by the services.

COCo Requirement 8: Provision of a standardized interface for incoming measurement data, presentation, and assessment processing

The presentation is related to the service but in a unified and service property independent manner. In the case of a service exchange, this helps to avoid adjustments in operations to changed circumstances. The process and service-oriented presentation simplifies the assignment of quality criteria.

COCo Requirement 9: Process- and service-oriented presentation of monitoring

Unlike the common system-oriented monitoring in client-server based systems, the monitoring has to be process- and service-oriented in SOA-based applications. In order to increase the overview in monitoring, an aggregation along the layers of SOA is useful. The found requirements are summarized in Table I.

In the context of monitoring, there are also quality criteria directly related to capacity management which are not closer analyzed in this contribution. However, investigations on that result in similar and additional requirements to the components of SOA. As shown for the processes of change/release management and test management, incident/problem management and monitoring and SLM, there is a strong need for another central component in SOA. COCo is needed to support the day to day operations and the usage of potential flexibility of SOA-based applications related to the opportunity of a service exchange.

TABLE I. SUMMARY OF REQUIREMENTS

Comp.	No	Requirement
S	1	Complete encouragement to fulfill the requirements to SR
	2	Standardized interface to change from one environment to another in the life cycle
	3	Service responsibility for the information on its interfaces stored in SR
	4	Provision of a switchable trace including service and activity information, time stamps, and resource consumption
	5	Forwarding of standardized exception information
	6	Standardization of the provided exception information independent on service technology
	7	A metric for the measuring key performance indicator
	8	Provision of a rule set for the assessment of measurement results (metric of relationships)
	9	Provision of measurement ranges for the assessment of measurement results
	10	Measurement of quality criteria during its execution
	11	Unified transfer of measurement data to the COCo
SR	1	Central management of service publication and contract
	2	Centralized Service Repository
	3	Information providing and usage to support operations
	4	Central information management for service life cycle management
	5	Management of relationships between repository objects
	6	Management assessment opportunities of service versions and planning
	7	Long term archiving of process and service models to assess dependencies between incidents and changes
	8	Administration of analyzable relations between functional and technical activities
	9	Integration of CMDB and SR to one central component
	10	Central SR to integrate and control the different statuses of the services in the environments along the life cycle.
	11	Storage for deployment interface information of services
COCo	1	Complete encouragement and analyzability of the information available in the SR
	2	Standardized interface for initiation of status and environment changes of services (deployment interface)
	3	Central switching of process trace including all services
	4	Provision of a standardized interface to services for the data transfer of trace pieces
	5	Build an ordered summarized central process trace on the basis of the transferred traces for each service
	6	Provision of a standardized interface for receiving exception data by services
	7	Recording of the order of the raised exceptions and assignment to services and processes
	8	Provision of a standardized interface for incoming measurement data, presentation, and assessment processing
	9	Process- and service-oriented presentation of monitoring
SB	1	Standardized adapter for the service deployment interface
	2	Preparation of input and output data in alignment with interface information in SR
	3	Simulation of data transfer from predecessor service and successor service aligned with interface descriptions for service stored in the SR

VI. DISCUSSION

Although SOA exists for several years and the evolution of this architecture lead to a common and often implemented infrastructure, our addressed challenges for operating such architectures are increasingly important for daily use. The issue of operating SOA is often addressed from a vendor perspective. In the case of

Buchwald et al. [18] investigate the requirements for SR to enable exchangeability of services in general. Nevertheless, operations are not the focus here.

A first approach for standardization to operating SOA is given by the standard for Application Response Measurement (ARM) 4.0 [29]. ARM proposes the integration of accordingly coding for response time measurement. They focus on the standardization of the interface ("How"), but do not take into account the "What". So it is not guaranteed that the measured values are comparable and assessable. With the usage of a service for operation, as we propose with COCo, a guarantee within the SOA operation can be given. This includes service exchange as well as incident or problem management. However, this requires a drastic change, not only in the operation phase of SOA, but also in the design phase and therefore, modelling aspects [30] have to take this into account.

VII. SUMMARY AND OUTLOOK

With examples of a number of very important IT service and support processes, we show that the general understanding of the components of a SOA has to be extended if the advantage of flexibility of SOA-based applications in the context of a possible service exchange should be achieved. If you really use the flexibility this will result in an increasing

effort in the operations of SOA based applications in keeping e.g. stability and availability as shown for selected processes in this contribution. By investigating in the other ITIL processes and the practical execution additional requirements will be found. COCo is required as the central operations platform independent on the service technology. This requires a unification of operations task within the COCo. This can only be fulfilled if especially services, as well as, SB, and SR support COCo. ITIL has shown that it is possible to describe best practices of the day to day IT service and support processes independent on any provider specifications. Nevertheless ITIL describes the good practices on a high process level. There is the same need for the practical approach of operations with a tool supporting this in SOA. We further present the general idea of such a central, standardized tool for the operations of SOA-based applications. We expect that for some of the new requirements general adapters can be developed which support every service. With help of the definition of standardized measurement criteria and the provision of decision rules as well as standardized follow up procedures are provided by the publishers in front and supported in general by COCo. Other requirements can only be fulfilled if established rules or a type of standard for non-functional properties exist. In this case, each service has to support these standards and these properties have to be already taken into account in the service design phase. If such a provider independent tool COCo can be established this will increase the usability of SOA and its flexibility by multiple. At the same time in is a step to become independent on the service providers because operations tasks are independent on the used technology and the service provider. The operations tasks become at least similar. To accomplish such a standard within a single company might be easy compared to the challenge of establishing an industry-wide standard incorporating providers like Oracle, IBM, SAP, or Microsoft. New challenges arise if one integrates services from a cloud into a solution what is not covered in our investigations.

ACKNOLEDGEMENT

The work in this paper has been funded in part by the German Federal Ministry of Education and Research (BMBF) through the Research Program "Digi-Dak+ Sicherheits-Forschungskolleg Digitale Formspuren" under Contract No.FKZ: 13N10818.

VIII. REFERENCES

[1] D. Sprott, "Business Flexibility through SOA", 2005, ftp://ftp.software.ibm.com/software/soa/pdf/CBDIWhitepaperBusinessFlexibiliyThroughSOA.pdf, accessed June, 5 2014.

[2] Software AG, Business Benefits, 2013, https://www.softwareag.com/corporate/solutions/soa/soa_solution/business_benefits/default.asp, accessed June, 5 2014.

[3] Office of Governance Commerce, IT Infrastructure Library V3: information technology infrastructure library version 3 core, The Stationary Office, London, 2007.

[4] Oasis, Reference Model for Service Oriented Architecture, 2006, http://docs.oasis-open.org/soa-rm/v1.0/, accessed June, 5 2014.

[5] I. Melzer, S. Eberhard, "Service-orientierte Architekturen mit Web Services: Konzepte - Standards - Praxis, 3rd ed., Spektrum Akad. Verl., Heidelberg, Deutschland, 2008.

[6] L. Will, "Requirements for Operations to Take Advantage of the Flexibility of SOA", Computer Science and Engineering 2(5): 68-76, 2012.

[7] D. Krafzig, K. Banke, D. Slama, "Enterprise SOA: Service-oriented architecture best practices", Prentice Hall Professional Technical Reference, Indianapolis IN, 2005.

[8] T. Erl, "SOA design patterns", Prentice Hall, Upper Saddle River, New Jersey, 2009.

[9] M.P. Papazoglou, W.-J. Heuvel, "Service oriented architectures: approaches, technologies and research issues", The VLDB Journal 16 (2007) pp.389–415.

[10] T. Erl, "Service-oriented architecture: Concepts, technology, and design", 9th print. ed., Prentice-Hall, Upper Saddle River, NJ, 2009.

[11] J.S. Evans, "Strategic Flexibility for high technology manoeuvres: A conceptual Framework", in Journal of Management Studies (1991) 69ff.

[12] F. Cummins, Building the agile enterprise: With SOA, BPM and MBM, MK/OMG Press/Elsevier, Amsterdam, Boston, 2009.

[13] L. Will, "Operations requirements in SOA based solutions", in: Fifth International Conference on Research Challenges in Information Science (RCIS), 2011, IEEE, Piscataway, New Jersey, USA, 2011, pp. 1–10.

[14] L. Teuber, C. Weidmann, L. Will, Monitoring and operations with SAP Solution Manager, 1st ed., Galileo Press, Boston, 2013.

[15] Oracle AG, Right from the Start: SOA Lifecycle Governance, 2012, http://www.oracle.com/us/technologies/soa/right-from-the-start-soa-1848279.pdf, accessed June, 5 2014.

[16] IBM Inc., Rational Quality Manager, http://www-01.ibm.com/software/rational/products/rqm/, accessed June, 5 2014.

[17] Oasis, "UDDI Version 3.0.2: UDDI Spec Technical Committee Draft, 2004, http://uddi.org/pubs/uddi-v3.0.2-20041019.pdf, accessed April, 7 2014.

[18] S. Buchwald, J. Tiedeken, T. Bauer, M. Reichert, "Anforderungen an ein Metamodell für SOA-Repositories", in: C. Gierds, J. Sürmeli (Eds.), 2. Zentral-europäischer Workshop über Services und ihre Komposition: ZEUS2010, 2010, pp. 17–24.

[19] M.P. Papazoglou, D. Georgakopoulos, "Service-Oriented Computing", in Communications of the ACM, 46 (2003) pp. 25–28.

[20] AXELOS, The Official ITIL site, http://www.itil-officialsite.com/SoftwareScheme/EndorsedSoftwareTools/Gold_Endorsed_Software_Tools.aspx, accessed June, 5 2014.

[21] Salesforce, "Change Sets and Deployment Tools", https://developer.salesforce.com/releases/release/Summer14/Change+Sets+Deployment+Tools, accessed June, 5 2014.

[22] SAP AG, "Transporting Non-ABAP Objects in Change and Transport System" http://help.sap.com/saphelp_nw70/helpdata/en/45/ec25370fdc3481e10000000a1553f6/frameset.htm , accessed June, 5 2014.

[23] A. Koesegi, R. Nerding, SAP Change and Transport Management, Galileo Press, Boston, 2006.

[24] A.-W. Scheer, "ARIS - Business process modeling", 3rd ed., Springer, Berlin, 2000.

[25] L. Ribarov, I. Manova, S. Ilieva, "Testing in a service-oriented world", in: International Conference on Information Technologies (Ed.), Info-Tech 2007 Proceedings Volume 2, 2007.

[26] Datanamic Solutions EV, Data Generator for Oracle, 2013, http://www.datanamic.com/datagenerator-for-oracle/index.html, accessed June, 5 2014.

[27] SAP AG, "SAP Test Data Migration Server", http://help.sap.com/saptdm, accessed June, 5 2014.

[28] L. Will, V. Köppen, "Zentrales, standardisiertes Monitoring als Grundlage des Service Level Managements in flexiblen SOA-Lösungen", in: U. Goltz, H.-D. Ehrich (Eds.), Informatik 2012, Ges. für Informatik, Bonn, 2012, pp. 759–773.

[29] The Open Group, "Application Response Measurement: ARM 4.1 version 1", 2007, https://collaboration.opengroup.org/tech/management/arm/, accessed June, 5 2014.

[30] M. Bell. "Service-Oriented Modeling – Service Analysis, Design, Architecture"