The implementation experience of an Advanced Service Repository for supporting Service-Oriented Architecture

Cleiton Garcia, Waldemar Roberti
Information Technology department
WEG S.A.
Jaraguá do Sul - SC, Brazil
cleitong@weg.net; roberti@weg.net

Daniel Schreiber, Marco Paludo,
Andreia Malucelli, Sheila Reinehr
Programa de Pós-Graduação em Informática (PPGIa)
Polytechnic School
Pontifícia Universidade Católica do Paraná (PUCPR)
Curitiba - PR, Brazil
xiraba@gmail.com; marco.paludo@pucpr.br;
malu@ppgia.pucpr.br; sheila.reinehr@pucpr.br

***Abstract* — To be effective, the systematic reuse in software development process must be supported by an artifact database. Although many organizations do not have such database, generally the Service Repository is the catalog responsible for leveraging Service-Oriented Architecture (SOA) reuse within the development lifecycle. This paper presents a SOA implementation experience, emphasizing the contributions of developing Service Repository extensions. The adoption of SOA occurred along with a new ERP deployment. One of the main goals was to enable the reuse of standard processes and services within a better integration between ERP system and legacy systems. The repository received additional capabilities to maximize searches and usability; to support configuration and release management; and also address performance issues within cache manageability. The benefits of this work are presented, as well as the results of the current situation in the organization considering SOA repository usage and the empirical evidence collected.**

Keywords-SOA; systematic reuse; service repository; uddi

I. INTRODUCTION

Many software development methods were proposed in order to leverage reuse. However, it is still a difficult goal to implement up to now. In the middle 80´s, the top down approach of the structured analysis was improved, reducing redundancy and incorporating event orientation [1]. The essential analysis had emphasized the identification of common points of functionalities and data structures, and the partitioning according to the modules behavior [2]. These were some of the first steps towards a reuse approach.

Nowadays, organizations are increasingly working to adopt a Service-Oriented Architecture (SOA) as a means of leveraging the reuse. In SOA approach, the reuse addresses the services level that maps business processes. The main goal of SOA architecture is to connect the business world to the Information Technology (IT) world, in order to make both more efficient [3].

The SOA is a style of architecture for integrating business processes and supporting IT infrastructure through standardized components used as services that can be reused and combined to answer the business changes [4]. This architecture is supported by a set of tools for achieving the reuse of process, components and infrastructure.

Before SOA advent, Software Product Lines (SPL) has worked to achieve the systematic reuse, exploring the similarities of the software components. According to [5], one of the major problems in SPL is establishing a repository containing the reusable software assets as well as the documentation for its use. As well in SPL, any software development life cycle concerned with reuse goals requires a repository.

SOA was introduced in 1996, so this architecture is not a recent subject [6]. The SOA projects conducted before the Web Services usage had resulted in low success rates, as presented in Figure 1. The main causes of failure were the complexity of technologies involved and the lack of knowledge to design the services in the right granularity for business [6].

After the advent of Web Services, SOA has become more powerful and with a great appeal for interoperability, compatibility with several programming languages, operating system independence, use of established standards, low cost of implementation, and easy to understand [7].

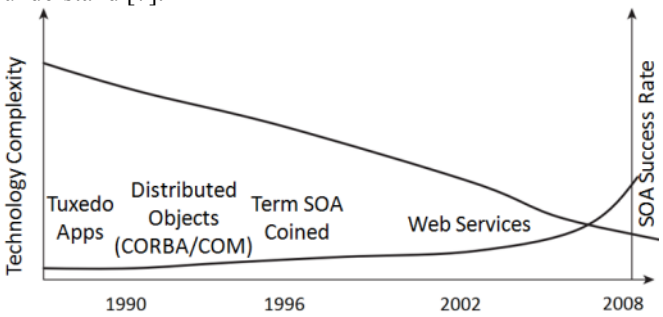

Figure 1. SOA time line [6]

The implantation of SOA architecture requires that both business and technical vision to walk together in continuous maintenance and evolution. When this visions don't still aligned, problems arise such as the services do not represent the organization's business processes, the conception of

services is focused only on the need of one consumer system, lack of adaptability, the development of services for peer-to-peer integration without analyzing the common points of connections of processes and consequently compromising reuse. These problems result in creating an architecture that is only a specific set of services [8], which is usually the result of a bottom-up approach of SOA implementation. For addressing this problem, a large number of software process lifecycle were proposed.

This paper presents the experience of SOA implementation in a large organization, with emphasis on extensions developed on the Service Repository for improving the service discovery and service publication. The paper is structured as follows: The next section presents concepts about Service Repositories; section 3 presents the research method used; section 4 describes the implementation scenario highlighting aspects related to the organization. The implemented solution, contributions to the asset repository, as well as discussion of the results, is presented in section 5. Finally, Section 6 concludes the work.

II. BACKGROUND

Several aspects related to the Service-Oriented development method have been discussed by academia and software industry. Since the conception of SOA, many deployment approaches appeared concerned with improving the planning, analysis, development and governance for supporting this architecture. Due to the diversity of software engineering methods applied to Service-Orientation, a guide to select methods according to the needs of the organization has been created. In this guide, characteristics of 12 methods have been evaluated [9]. This guide helps the organization to select one or combine some methods for addressing the gaps in the organization process.

Several software providers have their own methods for implementing SOA, so there are many ways to measure their maturity, making this subject controversial. Concerned in guide the organizations in the SOA adoption, Sonic Software, Systinet, AmberPoint, and BearingPoint established an alliance to create a single maturity model for SOA, known as SOAMM (SOA Maturity Model). This model was published in the OMG SOA SIG Forum, and it has maturity levels based on the CMMI - Capability Maturity Model Integration levels [10][11][12], elaborated by Software Engineering Institute. In a similar way of CMMI, the SOAMM was phased in consistent steps (levels) for achieving the SOA gains, as presented in Figure 2. [13].

In SOAMM model, as shown in Figure 2, level 1 has emphasis on training, pilot projects with Web site, Portal, custom integrations and construction of a small number of services. At level 2, the scope is extended to multiple systems, in order to institutionalize the use of SOA, and to solidify the architecture within the IT management and governance.

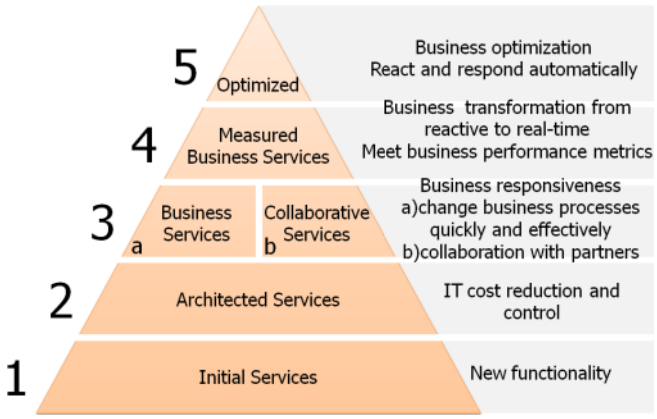

Figure 2. SOA maturity levels and corresponding benefits [13].

Similar to SOAMM, IBM has also defined a maturity model, called Service Integration Maturity Model (SIMM). This model was incorporated by The Open Group Service Integration Maturity Model, and nowadays it is called OSIMM [14]. Based on the combination of SOAMM and OSIMM, the Combined SOA Maturity Model was proposed in a previous work [15]. Currently the OSIMM was submitted to ISO under the publicly available specification process as a draft in ISO/IEC 16680.

At level 2 of SOAMM and level 4 of OSIMM, it is required the implementation of a common place for organizing the services catalog information including definitions, policies, and lifecycle management, which is considered an important step toward reuse [13], [14], [15]. The definition and implementation experience of this catalog is the main objective of this work with the some collected numbers of the production used along 3 years.

As presented in Figure 3. SOA provides a central component for recording the services, typically called Service Registry or Service Repository, where information about each service are stored, creating an inventory or catalog. The **first step** for using the repository is the publishing, when a new service is included in the catalog. In the **second step** the repository is used to discover, retrieve and reuse services, during design phase and also during execution time, however, it is most common during the design time [8].

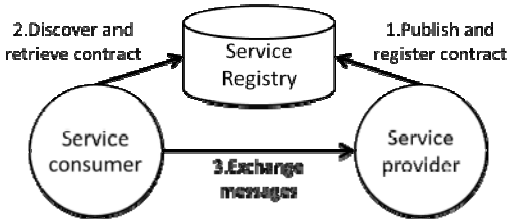

Figure 3. Message flow to publish and retrieve services [8].

The usefulness of the Service Repository can be related to consulting the yellow pages of a phone book, where queries can be performed to find a specialist who provides a specific service to solve a problem. On the other hand, the

persistence of a service occurs by recording the service provider organization (business entities), the description of the services, some details about service types (extensible structure called tModel), and relationships with business and subscribed consumers [16].

The Service Repositories are most used during the design phase, when it is necessary the information about the description and how and where the service can be executed, what information it contains, who is already consuming and even why it is being used [17]. After understanding the contract in second step the **third step,** in Figure 3. is the service execution. The last step is performed by exchange messages between provider and consumer systems.

By definition, the Web Service has a contract that defines the input and output messages format. Both messages are composed by data types, known as metadata or service contract. This contract is defined in an XML Schema (XSD). A Web Service is defined by the Web Services Description Language (WSDL), where these contracts are described. The WSDL is helpful to humans and machines understand how the service operations work. It is composed by 2 parts [18]:

- Abstract part defines the logical elements containing the definitions of the: **data types** used for composing the **message** contract; and the **portTypes** describing a set of operations related with the input and output messages;
- Concrete part defines the transport and physical access, composed by: **Binding** describes a concrete set of formats and protocols; and **Service** contains the relationships between portType and Binding describing a physical address called **endpoint**.

The main technology standards used in the Web Services are maintained by World Wide Web Consortium (W3C). The W3C is responsible for developing Web standards, like XML, XSLT, XSD and WSDL [16] [19].

The services contracts are stored inside the Service Repository, making the access centralized and easier. The Universal Discovery Description and Integration (UDDI) is a standard maintained by OASIS (Organization for the Advancement of Structured Information Standards), and one of the pillars of Web Services technologies [18]. The UDDI Business Registry (UBR) was a project that helps to confirm the efficiency of the UDDI specification. During 5 years, it has managed over 50,000 entries from Microsoft, IBM, and SAP. The UDDI data model is composed basically by following parts [18]:

- *businessEntity*: Information about the party or organizations who is publishing the service;
- *businessService*: Descriptive information about a particular family of technical services;
- *bindingTemplate*: Technical information about service entry point construction specifications;
- tModel: Descriptions of specifications for services or taxonomies. It has various uses in the UDDI registry and offers an extensible structure to define other information.

During the service publication, first step, the WSDL is published in the Service Registry. At this step the tool should automate the creation performing a conversion from the WSDL structure to the UDDI data model. The Figure 4. describes the common mapping between WSDL and UDDI V2 or V3 data model [18]. In this work this conversion was extended to include new information and logging all modification in the description of the services.

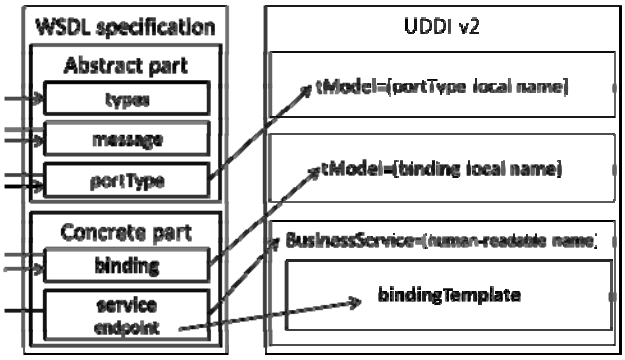

Figure 4. Mapping between WSDL and UDDI [18].

The Service Repository usually integrates with UDDI. However, The Service Registry is usually associated with real-time execution, but some references use Service Registry for both, Registry and Repository [8], which shows that there are still open concepts with lack of standardization.

A related work proposed three guidelines for avoiding the recurrent problems in the provider and consumer application engineering. The guidelines were organized for each one of the 3 steps explained in the Figure 3. . The first step is divided in two sub-guidelines, according to the construction approaches: contract-first, where the service realization is oriented by the design of WSDL, and code-first, where the Java, .NET or C++ codification generates the contract dynamically [20]. This work extends these previous guidelines with a real SOA implementation and the development of a tool for improving the service publication and service discovery process. This improvement was achieved through adding extensions attributes for defining detailed information about the services, extending the searches results, supporting cache capabilities to deliver higher performance, and operation process such as configuration and release management. This tool was constructed along the development cycles according the methodology in the section III.

III. METHODOLOGY

The work was based on the method action research [21], considering:

- The research focus on research in action rather than research about action: researchers acts to solve organizational problems along with those who live the problem;
- Participative research: researchers are external agents and act facilitating and supporting discussions and reflections about the research; internal people acts evaluating the results;
- Sequence of events: iterative cycles to gathering data, feeding them back to those concerned, analyzing the data planning action, taking action and evaluating the results.
- An approach to problem solving: application of a scientific method of fact finding and experimentation.

The authors of this work are part of the team that faces the daily problems concerning software development. The definition of the solution and evaluation of the results are conducted through suggestion of the internal team and numbers collected along the years of productive uses. The results are not based in experimental study; they were collected from the productive environment.

Through the action research, iterative cycle were conducted to find solutions, as shown in Figure 5. This cycle starts with a pre-step called Context and Purpose that aims at defining why is the project necessary and what are the forces driving the action.

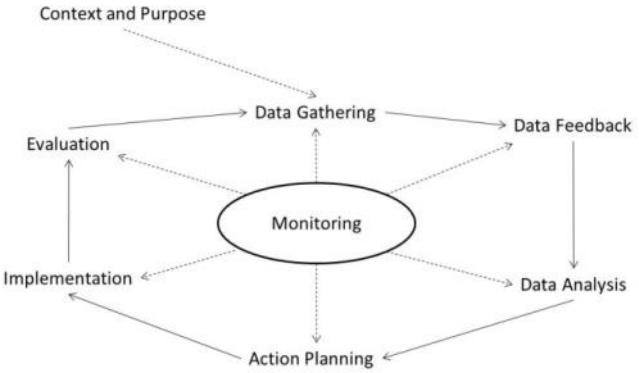

Figure 5. Action research [21].

The next six steps are related to the research itself. Three of them are related to data (gathering, feedback, and analysis) and three are related to action (planning, implementation and evaluation).

The next sections describe the context of the research that was carried out using the action research methodology.

IV. CONTEXT OF THE WORK

The project was implemented at WEG, which is an organization that operates on energy solutions, electric motor production, power generation and distribution, process automation and industrial paints. WEG is a Brazilian Manufacturing that has over 23,800 employees [22]. During the ERP deployment project that took about 2 years to be concluded, the implementation of new business processes has occurred. The new processes had to connect the global presence purchasing, producing, and selling anywhere in the world, improving the visibility of the production, warehouses, sales representatives, distributors, service network, customers and suppliers.

The ERP implementation project has led to redesign several business processes. According to Accelerated SAP Methodology (ASAP), at the Business Blueprint phase (similar to the elaboration in RUP), all the processes were mapped, designed or redesigned and classified [23]. This classification divided the WEG processes into levels which were applied to identify each IT assets, such as systems, services and components. The IT teams are also organized according the first classification level called macro-processes, and is presented below:

- Sales and Marketing
- Product Development
- Supply Chain Management
- Production Management
- Project Management
- Financial Accounting and Controlling Management
- People Management
- Management of Quality and Social-Environmental Sustainability
- Information Technology Management
- Corporate Management

The detailed flow of activities is on the third and final level, called sub-process, where the modeling is performed using BPMN (Business Process Modeling Notation). This modeling is supported by Aris [24] tool, which is compatible with BPMN, accelerating business processes automation, transformation and monitoring.

WEG is composed by several companies with systems development departments spread around the world. This characteristic leverages communication problems related to IT projects, so the communication management demands constant attention. These teams in different time zone need to know the assets of the Enterprise Architecture.

The repository was one of the possible answers for this problem, enabling access to information anytime. Before the new ERP adoption, the group had 583 legacy systems, but among them, just 72 had remained integrated with the ERP. The systems that were integrated into the ERP have different responsible teams, along with third party systems. The control and management of the architecture are complex tasks, arising a good opportunity for SOA.

V. SCHINDLER REPOSITORY

Considering the improvement opportunities in the service repository, researches results, lived problems in the organization, and the requirements not accomplished by market solutions, a new project and, consequently, a new tool was developed, to facilitate the management, persistence, collaboration and search of services and systems available in catalog. This tool was called Schindler, because besides cataloging services, Schindler is an inventory of systems that centralizes information concerning the lifecycle, deactivation schedule according to a cutover plan, maintenance dates and service level agreements.

The SOA maturity models were the sources for these requirements. The Open Group SOA Integration Maturity Model [14] (OSIMM) defines a roadmap for incremental steps for SOA adoption, maximizing the business benefits. On OSIMM level 5 it is required that the organization has a Business Data Dictionary & Repository. On level 7, the entire organization should be able to access and query this repository using information or characteristics about the service, contract interfaces or process information [14].

During the ERP implementation project, all captured and relevant information about the integration, services and systems were maintained on Schindler, which have been widely used. So, this tool is a Service and System Repository that was developed and deployed in productive environment and has supported the IT.

The Schindler repository is based on a generic database designed to store information of any component, which can be a service, system, software component or a server. Schindler replicates part of the services information on the UDDI. Due to the UDDI standardization carried out by OASIS, it was even possible to easily migrate between the UDDI solutions, what was proved worthy during performance evaluation and migration to version 3.0.

The use case diagram, presented in Figure 6, shows the scope and summarizes the functionalities supported by Schindler. The UC1-Browse the components, for example, was extended by other use cases focused on Systems, Services, Components and Servers.

The Figure 7. presents an example of one of the use cases (Browse services) where services can be found using free text or using top boxes filters. The three filter boxes have the information about the business process, contract, and related systems. Information about the provider and the consumer systems have been included for each service, in order to integrate system data and involved services, as well as the execution addresses of each service (endpoints).

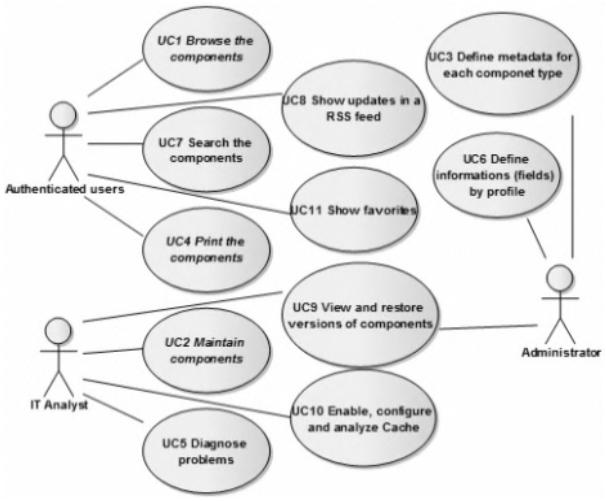

Figure 6. Schindler use cases.

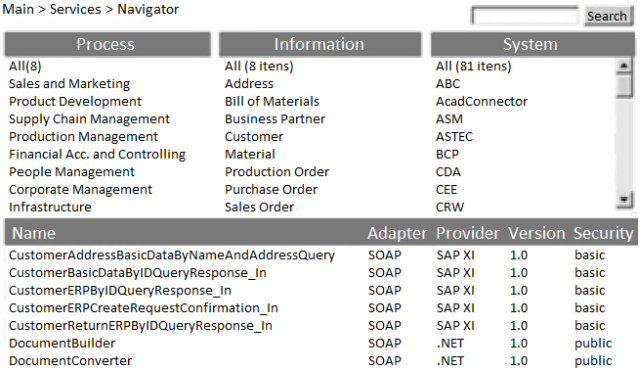

Figure 7. User Interface for UC1.1- Browse Services.

A. Services Configuration Management

The Schindler and service clients (or proxy) contribute through a specific configuration mechanism considering selected environment and usage scenario. Generally, the Service Repositories have information and independent configurations of all the different environments that support the lifecycle development. These organization's environments are usually called sandbox, development, quality assurance (QA), test, training, and production. The number of environments is proportional to the work for maintaining these configurations for each environment.

To simplify this configuration, it was proposed to maintain a unique and central environment, where the service is configured by defining different execution points according to each environment. The endpoints for each environment are maintained by an extension in the tModel structure, like the extension proposed by [25]. A previous work extended the UDDI structure to store the policies of the service consumers, focused in the context Quality of Service Management for Web Services on Business Process Management (BPM) Systems [25].

Before the service execution, the consumer performs a query on the UDDI to discover what is the endpoint according to the target environment and the usage scenario, identified by the consumer system name. Figure 8. shows the bottom part of the UI (user interface) to register services, where the execution addresses are defined by each environment and consumer systems.

Figure 8. Services execution addresses by environment and consumer.

Due to the use of the UDDI at execution time, it is possible to dynamically make changes in the endpoint addresses without changing the consumer systems. Through this flexibility it is possible to: define usage metrics, allocate the consumer systems to a specific server in the cluster (or channel in the Enterprise Application Integration), and control the traffic and cost distribution. These tasks help the problem identification, both business problems and services defects. All tasks performed without affecting the systems execution and without the need for building and releasing a new version of the system package.

B. Services search improvements and reuse results

The Schindler full text search uses common information as service name, description, and considers the information about several related objects. This variety of information helps the user to find a larger set of services, aiming to promote identification and, as a result, the reuse. All information is indexed by the developed search engine in order to answer quickly to the user in a full text search. These contexts information related with each component are following presented:

- The organization's business processes;
- Providers and consumers systems and modules;
- Message contract composed by the data types;
- The consumed services, usually when the service is composed by other services;
- Name of the server of execution as Domain Name System (DNS) or Internet Protocol (IP);
- Teams or people responsible for the services, in order to be accessible for communication.

Due to the additional information, Schindler can support root cause analysis of integration problems. Although unintentional, this new feature has aggregated new value to the tool, helping the operations support team. Before the implementation, system integration problems were usually transferred to the development team. After the Schindler implementation, this team became able to analyze the dependencies between system integrations and also to identify the destination addresses used by the consumer systems of each service and the related server.

It was also identified the opportunity to facilitate contact between the teams that have developed the service and the teams that will consume the service. Thus, the information about the creators of the services was persisted, including the responsible team, analysts and developers.

The previous integration architecture based on data layer did not contribute to achieve reuse. For each new integration the reading or writing routines were always rewritten in each of the consumer systems. In other words, the same database query was written many times. In a short time, there were dozens of systems with replicated functionalities (specifications and source code) to deal with the business entities. The impact of replication was not only a waste of resources, but was mainly a barrier for business process improvements. The software maintenance often required changes in several parts of the code that accessed the same tables on database. The lack of change traceability in these business entities was a frequent problem that caused improvements failures. The Schindler tool enabled analyzing the services reuse results for all systems, ERP and legacy.

As shown in TABLE I. , it is possible to observe the number of services, as well as the increase in the last developed systems in 2010 and 2011. The last system presented in TABLE I. , Maintenance Order Confirmation, is a composite application which combines 16 Web Services in a specialized UI, aiming at improving user productivity. An interesting point to analyze is that this system was developed using only existing services, without necessity to create new services. The rapid construction of this product was partly due to the pre-existent services, the effort to search the asset, a reuse activity facilitated by the Schindler repository.

TABLE I. AVALILABLE SERVICES AND NEW SERVICES BY SYSTEM.

Systems	Development hours	Used Services	New Services
Management Essays	5000	16	2
Electro mechanics Specification	5000	16	5
Guarantees	1200	23	7
Energetic Efficiency Calculation	2000	12	1
Manufacturing Execution	3000	11	1
Maintenance Confirmation	500	16	0

Taking into account just the five services with more consumer systems, they have saved more than 285 KLOC (Thousand Lines of Code), as detailed in TABLE II. The reuse is the main responsible for saving efforts and resources to produce and maintain all these lines of code. Independently of the SOA reuse approach, the goals could be achieved through other systematic reuse approach.

However, in the studied organization the systematic reuse was not occurring before the SOA implementation.

TABLE II. TOP 5 CONSUMED SERVICES.

Service	Consumer Systems (CS)	LOC	Saved LOC (LOC * (CS -1))
Documents	23	4,878	107,316
Materials	26	2,889	72,225
Bill of materials	18	3,122	53,074
Classification	24	1,624	37,352
Business Partners	10	2,691	24,219

In comparison with integrations based on data replication, the benefits with service approach are relevant concerning the storage space. For example, the information about document management in the ERP environment requires more than 2 Tera Bytes and the material master data use half Tera Bytes that hinders replication approaches. As explained before, the data replication approach does not encourage the reuse of business rule using the database platform. Additionally, the data replication requires a larger investment to maintain the replicated data entities more infrastructure for storage and processing resources to execute the replication routines.

The legacy systems quality is gradually improving, as it can be observed in the defects per KLOC metric of the IT department. This reduction has saved efforts in maintaining the software lifecycle. These positive results are not just because of the adoption of reuse and SOA initiatives, but a set of software engineering initiatives, such as automated unit and functional tests, quality control metrics, continuous integration with verification of code design metrics for Java, C#, VB.NET, and ABAP source code.

Reuse should also exist at the service contract level, making a data type common for a range of services and even for the entire organization and partners. The analysis of business information should standardize the semantics for the business entities, defining a common vocabulary. There are several organizations that specify standards for several areas of industry, e.g. for the automotive industry (AIAG), healthcare (HL7), Chemical (CIDX), banks (BIAN, SWIFT, UNIFI), trade (RosettaNet, 1SYNC), among others. This standardization leverages the communication of the IT department and system users, and it is also reduces the effort for systems integration, as soon as the software suppliers adhere these standards. The service artifacts (contracts and glossary) produced by different projects can consume and feed the centralized glossary reducing efforts and costs.

C. Performance Improvements

A problem faced after rebuilding the integration was the performance of the Web Services. Before the SOA implementation, the integration usually occurred usually on the database layer, using views, synonyms or triggers. This approach offered a great performance when reading and writing information in a centralized database. In distributed architectures, challenges are raised such as communication between servers, many data repositories and connections involved, amount of messages exchanged, protocols overload (HTTP, SOAP, WS-Security, WS-RM), volume transported within XML messages, etc.

It was implemented a message cache solution that improves the performance of services and reduces the workload on the provider systems. The solution consists in indexing request message. The request and response message are maintained for reuse by the next requests. The cache indexes the request messages through a list of short key in order to accelerate the searching. Each short key is building applying the classical SHA512 algorithm. The SHA512 is a cryptography function usually used as: a security standard for many organizations, governments, and also used by common user in the error detection during transmission of data (called as checksum).

When the cache does not have the information, it request to the provider system. After receive the response from the provider system, the cache answer it directly to the consumer system, and finally persist the information in the cache on an asynchronous process.

Due to security reasons, the cache checks the authentication in the HTTP header or in SOAP header (WS-Security), before finding out in the hash. It was also created an integrated monitoring using the Kerberos protocol [26]. This monitor can filter and select all information used in the cache, check the data in the request and response messages, and also is the provider of information in the TABLE III. , TABLE IV. and Figure 9.

When registering a service in Schindler, it is possible to enable the cache and set a retention time for the messages. This time represents the maximum time that the message can be kept in the server cache. When the cache is enabled, the default value defined was one day. This initial configuration produces the visualization of changes until 24 hours. Therefore, it became necessary to develop triggers to execute the cache refresh in case of immediate updates.

According to next tables, it is possible to analyze the efficiency statistics, where the cache is being more effective. These data represent the real-time performance of the cache. How this reports is real-time, the information varies according to the number of users, time of the day, memory capacity, algorithm for discarding, retention time, among others factors. The TABLE III. presents the most requested services and cache **Efficiency** calculated through the **Cache Hit** Count divided by **Total Requests** Count. It is possible check that with only few requests to the provider service is it possible answer to thousands requests and users with the same information maintained in the cache memory. It usually occurs because certain information is strongly used by a large quantity of users, than with a relatively small quantity of data in the cache is possible answer thousands of request.

TABLE III. SERVICE PERFORMANCE AND EFFICIENCY IMPROVEMENT.

Service	Total Requests	Cache Hit	Provider Requests	Efficiency (%)
Master data changes	3,794	3,785	9	99.76
Characteristics	33,298	33,201	97	99.71
Material Classification	4,419	4,389	30	99.32
Documents	5,876	5,787	89	98.49
Materials	2,577	2,182	395	84.67
Partners	1,396	1,104	292	79.08
Bill of materials	2,182	1,537	645	70.44

The TABLE IV. presents for each service: the average size used for cached messages where the average of request and response message size are summed; the worst time for receiving a response from the provider system; the average time to call the provider system, and the total average time to cache answer. This last information takes into account both kind of message exchange: using the data in cache memory and also the directly requests to the provider service. According the information is possible analyze that the total average time between: 112 times faster for partners and 5,750 times faster for characteristic. The characteristic service has extremely high numbers because the intensive use of the characteristic for the product development and production.

TABLE IV. SERVICE PERFORMANCE AND AVERAGE SIZE AND TIMES

Service	Message Average size (KB)	Provider worst time (s)	Provider average time (s)	Total average time (s)
Master data changes	16	2.41	1.54	0.0029
Characteristics	464	23.77	2.30	0.0004
Material Classification	423	10.80	2.18	0.0014
Documents	97	80.49	13.72	0.0839
Materials	23	24.39	1.48	0.0006
Partners	302	24.14	1.84	0.0163
Bill of materials	25	26.14	1.98	0.0009

The main reason for implementing the cache was the performance needs, but some service providers are billed by the data volume. In this billing method, the return of investment is proportional to the amount of data answered by the cache. After the implementation of the cache solution, the data volume communications were significantly reduced. As presented in Figure 9. , in the first quarter of use, without the cache, all requests were made directly to the provider, usually the ERP. After the cache, the volume of Giga Bytes to the provider was significantly reduced. In the first month, the total volume of use was 152.60 Giga Bytes (GB), and the calls to the provider were 53.76 GB, resulting in a reduction of 98.85 GB. Approximately two-thirds of the requests were answered directly and quickly by the cache memory.

In 2011, the monthly average total volume was about 339.22 GB and the request to the provider was about 42.52 GB, resulting in an averaged of reduction of 296,70 GB. It results in a total efficiency of 87.47%. This efficiency was results of a set of actions, such as: use cache only for the right services, it is mandatory to know the update life cycle for each data entities in the organization, increase the RAM memory in the server it is much faster, adjusts in the retention time, among others. It is important noting that the variation is influenced by other business factors, such as internal projects, market situation and selling volume changes, and vacation seasons.

With the same purpose, addressing performance issues, a second cycle of the research action showed the opportunity for accelerating the service recovering. At this step the UDDI inquiry Application Programming Interface (API) is used to discover the execution address. Considering the proposed tModel extension structure presented in the point A, the discovery time was taking between 0,2 and 0,4 seconds for two market solutions and one free. For this reason, it was indexed in memory the UDDI data model containing all tModel structure. It achieves quicker queries to consumer system discovers the execution destination without performance degradation.

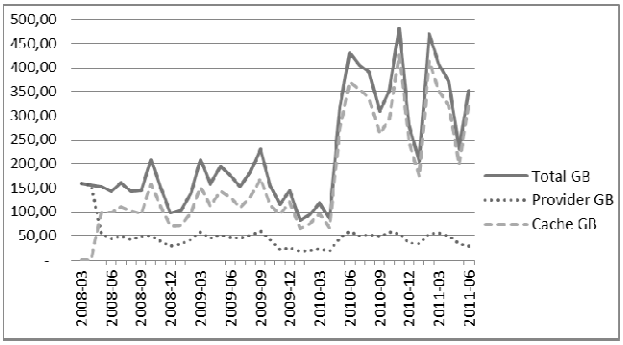

Figure 9. Data traffic reduction.

D. *Publishing assets to Service Repository*

As earlier described, the Service Repository is used in the design phase and also in the execution time, where the consumer application requests the endpoint in runtime. Considering that the current scenario includes a single Service Repository, it is necessary to have a robust change management. This robustness is mainly to avoid that any change in the development or test (quality assurance) environment causes any impact in the productive environment. The systems are operated by the users in the productive environment changes in the service repository can affect the availability and according the solution criticality can causes unpredictable financial losses.

As can be observed in the activity diagram in Figure 10. , the process of saving a service can be performed in two ways: a way called off-line mode, that only updates the generic component database without updating the information that are used at runtime (UDDI), and the second way, called on-line, where information are updated in both repositories. When the information is saved in the UDDI, it is mandatory to use persist in the UDDI data model, as described in the section 2 with the extension in the section 5 point A.

Every time that a Schindler user saves a component, the tool checks the component existence in the generic database. When the component already exists then it is created a new revision and the previous configuration is not changed in the database. This configuration logging is maintained aimed to support the problem identification.

Alternatively, it is possible to save an off-line configuration (only in the generic database) and afterwards perform only the process of activating this configuration in the UDDI (replicating the information from database to the UDDI). As well as it is also possible to restore a previous version of the services configuration. After any changes in the services registry, the information is provided in a RSS channel in order to improve the communication among the developer teams.

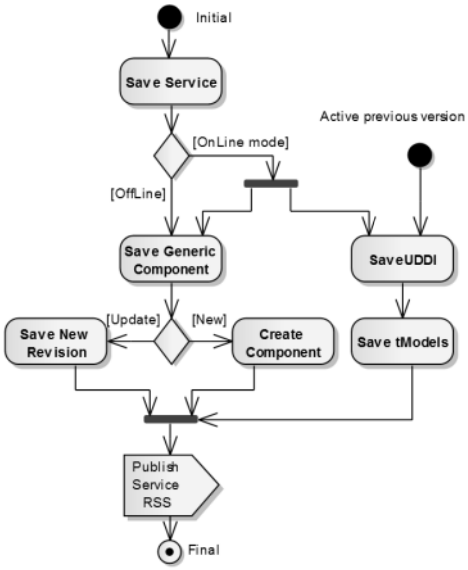

Figure 10. Service Registry activity diagram.

Publishing the changes into the production environment is a critical process that requires audit records in order to identify all changes. To accomplish this requirement it was developed a history logging of the configuration versions. This history allows visualization of the changes and applying filters by date, user and request number. To complement the previous requirement, it was also developed a feature that accelerates the process of restoring a previous configuration version.

E. Institutionalization of SOA and BPM method

An important step for the organization is establishing the software process, including the reuse of services, components, documentation artifacts and infrastructure. In order to achieve this objective, the organization needs to include reuse practices as formal activities exploited on modeling the software process. The IBM proposed a software development life-cycle method for designing and building SOA-based solutions, called Service-Oriented Modeling and Architecture (SOMA) [27].

In a similar way of SOMA, the organization's software development process was organized in phases based on a central phase focused on identification of assets. This identification phase is also based on the AI1 Identify Automated Solutions process from COBIT Framework, where the objective is to minimize the cost to acquire and implement solutions whilst ensuring that they enable the business to achieve its objectives [28][29].

Theses phases presented in the Figure 11. guide the set of software process. The business modeling phase is optional, according the project size and organization impact, followed by the identification phase which guides the decision to buy or make solution. After the identification phase, iterations of the specification, construction and deployment phases are done until the product is delivered.

The SOMA is a convenient choice due to the maturity of its use in real industry applications and the existence of several guidelines. Other important point was the integration with Rational Unified Process (RUP), object-oriented analysis and design (OOAD) and business processes management (BPM), according to previous studies for evaluating the existing Service-Oriented Software Engineering (SOSE) methodologies [9]. These integrations help to involve organization stakeholders which are running similar methods with Open Unified Process (Open UP) or RUP to effectively make the transition.

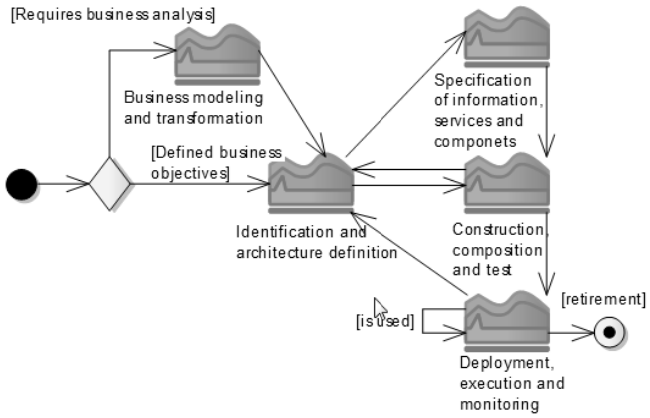

Figure 11. Institucionalization of SOA and BPM method [23].

VI. CONCLUSIONS

Defining a repository is an important step towards systematic reuse in organizations software processes. The Schindler tool has been supporting the IT users and the searching feature plays a very important role making this practice feasible. This work is different from previous works due to the improvements in the extension on the repository information and using it in full text searches, confirmed by results in the industry. The integrated cache helps to answer to SOA problems, as well as the publishing approach supports the change release on productive environments.

Service Repositories could add greater value when including more information about the business, because a service is often used in different contexts and maintaining

these relationships is a major challenge. For future works, BPM brings artifacts that offer possibilities to improve the ability to semantic search. The existent artifacts are, for example, the Business Process Modeling Notation (BPMN) and Business Process Execution Language (BPEL).

The advent of Representative State Transfer (REST) opens opportunities to extend the Service Repository. This new category of Web Service uses only HTTP protocol without SOAP, minimizing the communication overhead. However, it needs more standardization and support for descriptions artifacts, where the classical WSDL was replaced by the WADL. The repository needs new mapping rules to automate the registry of application services through the WADL.

The need to rebuild all the ERP integrations has shown a great opportunity toward implementing the SOA architecture. According to the results up to present days, it is possible to conclude that the Schindler has been showing its value in the software development life cycle and leveraging the software reuse. The maintenance and change management processes are been supported, but there are some opportunities for improvement, such as supporting other infrastructure assets, service contracts, licenses, integration with Configuration Management Database (CMDB) and monitoring tools.

ACKNOWLEDGMENT

The authors thank to the Universidade Católica do Paraná for supporting this study. They also thank the Católica de Santa Catarina and WEG for the financial supporting. They also thank WEG for allowing the research project in partnership with academia, and the co-workers André Vinícius Castoldi, Darley Rodrigo Machado, Flávio Kannenberg, Jackson Donadel and Kleber Kiefer who collaborated on the development and deployment.

REFERENCES

[1] E. Yourdon, *Modern Structured Analysis*, Englewood Cliffs: Prentice Hall, 1989.

[2] S.M. McMenamim, J.F. Palmer, *Essential systems analysis.* Englewood Cliffs, Prentice-Hall, 1984, 567p.

[3] W. A. Brown, , R. G. Laird, C. Gee, T. Mitra, *SOA Governance: Achieving and Sustaining Business and IT Agility*. Indianapolis, IBM Press, 2009.

[4] N. Bieberstein, R. Laird., K. Jones, T. Mitra, "Executing SOA: A Practical Guide for the Service-Oriented Architect", Indianapolis: IBM Press, 2008.

[5] T. Käkölä and J. C. Dueñas, *Software Product Lines: Research Issues in Engineering and Management*. Springer, Berlin Heidelberg New York, 2006.

[6] M. Rosen, B. Lublinsky, K.T. Smith, M. J. Balcer, *Applied SOA: Service-Oriented Architecture and Design Strategies*. Indianapolis, Wiley Publishing, Inc., 2008.

[7] G. Hohpe, B. Woolf, *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Boston, Pearson Education, 2004.

[8] T. Erl, *SOA: Principles of Service Design*. Vancouver, Prentice Hall, 2007.

[9] Q. Gu and P. Lago, "Guiding the selection of service-oriented software engineering methodologies," Service Oriented Computing and Applications, pp. 1-21, 2011.

[10] Software Engineering Institute. CMMI® for Development (CMU/SEI-2010-TR-033), Version 1.3. Pittsburg: Carnegie Mellon University, p. 482, 2010.

[11] Software Engineering Institute. CMMI® for Acquisition (ESC-TR-2010-032), Version 1.3. Pittsburg: Carnegie Mellon University, p. 438 2010.

[12] Software Engineering Institute. CMMI® for Services (CMU/SEI-2010-TR-034), Version 1.3. Pittsburg: Carnegie Mellon University, p. 520, 2010.

[13] SOAMM, A new Service-Oriented Architecture (SOA) Maturity Model, 2005. Avaliable: <http://soa.omg.org/Uploaded%20Docs/SOA/SOA_Maturity.pdf> (accessed Jun. 2011).

[14] Open Group, Service Integration Maturity Model Technical Standard: The SOA Source Book: The Open Group, 2009. Avaliable:http://www.opengroup.org/soa/source-book/osimm/model.htm (accessed Apr. 2012)

[15] E. Söderström and F. Meier, Combined SOA Maturity Model (CSOAMM): Towards a Guide for SOA Adoption, *Enterprise Interoperability II*. London, Springer, 2007, pp. 389-400, doi: 10.1007/978-1-84628-858-6_43.

[16] T. Erl, *Service-Oriented Architecture: A Field Guide to Integration XML and Web Services*. New Jersey, Prentice Hall, 2008.

[17] S. Carter, 2007. *The New Language of Business: SOA & Web 2.0.* New Jersey, Prentice Hall.

[18] OASIS, Advancing open standards for the information society. Available: http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-wsdl-v2.htm (accessed Apr. 2012).

[19] W3C, Extensible Markup Language (XML). Available: http://www.w3.org/XML/ (accessed Apr. 2012)

[20] M. Crasso, C. Mateos, A. Zunino, and M. Campo. The EasySOC Project: "A Rich Catalog of Best Practices for Developing Web Service Applications", CLEI Electronic Journal, vol. 14, no. 3, pp 33-42, 2011.

[21] P. Coughlan, and D. Coghlan, D, Action research for operations management, *International Journal of Operations and Production Management*, 2002, v.22, n.2. Academic Research Library, 2002, p. 220-240

[22] WEG. "History – The Founders", Avaliable: http://www.weg.net/us/About-WEG/History/The-Founders (accessed Jun. 2012).

[23] V. KALE. Implementing SAP R/3: The Guide for Business and Technology Managers. Indianapolis -USA, SAMS - A Division of Macmillan USA.

[24] ARIS Platform, Available: <http://www.softwareag.com/br/products/aris_platform/default.asp>, 2011. (accessed Apr. 2012).

[25] D. Z. G. Garcia and M. B. F. Toledo, A UDDI Extension for Business Process Management Systems. *Proceedings IADIS International Conference WWW/Internet*. Vila Real, Portugal, 2007, pp. 109-116.

[26] B. C. Neuman and T. Ts'o, Kerberos: An Authentication Service for Computer Networks, *IEEE Communications*, 1994, doi: 10.1109/35.312841.

[27] A. Arsanjani, S. Ghosh, A. Allam, T. Abdollah, S. Gariapathy, and K. Holley. 2008. SOMA: a method for developing service-oriented solutions. IBM Systems Journal. v.47, n.3, 2008, p. 377-396, doi:10.1147/sj.473.0377.

[28] ITGI, *COBIT v4.1*, Information Technology Governance Institute, Rolling Meadows, 2007.

[29] C. Ott, A. Korthaus, T. Boehmann, M. Rosemann, H. Krcmar: "Towards a Reference Model for SOA Governance", in: P. Soffer and E. Proper (Eds.): CAiSE Forum 2010, Hammamet, Tunisia, 09-11 June 2010, http://ceur-ws.org/Vol-592/Paper04.pdf.