Serdica J. Computing **6** (2012), 267–286

Serdica
Journal of Computing

Bulgarian Academy of Sciences
Institute of Mathematics and Informatics

APPLICATION OF SERVICE-ORIENTED ARCHITECTURE IN SOFTWARE QUALITY MANAGEMENT

Olga Marinova

Abstract. This article examines the quality management software processes and offers a model for their automation based on Service-oriented Architecture. The prerequisites for creating such a solution are analyzed, as are existing automated tools in this area. The possibilities of service-oriented architecture are presented along with its advantages in the context of this research on developing a quality management system that will operate effectively against defined requirements, consistently and regardless of the used platform, database management system (DBMS) and other technological features of the applications.

1. Introduction. In recent years of dynamic and volatile markets, the interest in quality is driven by people's rising expectations towards products and services, and also by more vigorous competition. In the context of software organizations, each developed product must meet the increasingly stringent requirements to match quality parameters expected by the user.

The increasing complexity and rapid development of information technology, as well as the boom in the variety of software solutions, require continued

ACM Computing Classification System (1998): D.2.5, D.2.9, D.2.11.
Key words: Service-oriented architecture, software quality management, quality and test management tools, application lifecycle management, software quality, services.

attention and quality assessment, even at the stage of software product development. In reality, the evolution of software development poses a specific threat to its effectiveness and expected behavior, which enforces broad-scale research of the relationships between different aspects of software quality. In order to investigate these relationships, the meanings of quality should be differentiated so that relative or quantitative evaluation can be achieved. For its part, assessing and measuring software quality requires defining and measuring its characteristics. After that effective mechanisms for tracking and controlling the quality criteria, based on defined quality factors, should be provided. Today more and more companies realize this need and implement automated tools both for testing the software during its development, and for measuring and managing the software quality.

The purpose of this paper is to analyze existing tools for automated software quality management and to propose a new model of information services for the software quality management processes through Service-oriented Architecture building.

Software quality management is a complex process that depends largely on the successful implementation of almost all processes comprising the building of a project. These include tracking the status and project progress, controlling the implementation of quality criteria, risk management, planning and executing different tests, defect management, configuration management and control. Each of them affects the quality assurance of software and is responsible for achieving certain goals and requirements for quality guarantee, according to the planned budget and time.

The paper is structured as follows. In section 2 the existing level of automated tools that are applied to manage software quality is presented through analyzing some of the most popular integrated packages for test management and partial quality management. Section 3 specifies the prerequisites for building a unified software quality management system and the main components that such a system should include. Section 4 describes in detail the application of the SOA architecture as a basic concept for the integrated system for managing software quality proposed in section 3. Section 5 (Conclusion) presents concluding remarks on the described integrated approach to quality management based on SOA architecture and the potential risks associated with its implementation.

2. Tools for software quality management. According to Gartner Research, the automated software quality assurance market is a subsegment of the overall **ALM (Application lifecycle management)** market [6]. They also note that software quality covers many more activities than before and that the leaders in this segment have begun rapidly expanding the set of tools to include their complex packets in order to achieve better integration with the whole development lifecycle.

In other words, the current focus on providing tools to automate various testing activities is shifting to more fully capture the processes that make up the quality management.

In support of this statement is the survey by Forrester Research [13], according to which, chronologically, software companies were initially oriented to quality assurance—in the 80s and 90s of the last century, then to quality control—in the late 90s and early 21st century, and nowadays and most likely in the near future the focus is on *software quality management*. This determines the appearance of many tools that offer different types of automated testing, and some of them even ensure effective management and quality assurance. Before analyzing their capabilities, we shall note some of the most important prerequisites for using automated tools to manage software quality:

- possibility to manage the requirements by defining the system requirements and to track the degree of their implementation in real-time, controlling the related defects and determining the strategy for risk management;
- reporting, tracking and analyzing key quality indicators;
- providing tools to monitor processes, tasks and project development, and the possibility of alerts and warnings in case of inconsistencies, delays or deviations from the chosen strategy and tasks;
- possibility for an automated management of the project—preparing a project plan, tracking the implementation of individual project tasks and assessing their compliance with the planned requirements, monitoring and identifying signs of potential risks;
- possibility to reuse and implement existing test assets on different versions of the product;
- implementing a large amount of tests more frequently;
- implementing tests that are complicated or even impossible to fully perform by hand, such as measuring performance, code coverage, etc.;
- automated testing tools provide more reliable methods for debugging and analyzing results;
- automation contributes to a more rational use of resources—it is a powerful tool for saving time and human resources.

Thus it follows that the automation issues of key quality management processes are becoming more popular and as a result many business leaders in software engineering have started working towards the development of integrated software solutions in this area.

Most software packages for quality management offered today provide tools for automated testing that can be categorized into three major groups—**Test Management**; **Functional and Regression Testing tools** and **Performance and Load Testing tools**. Some of the more popular tools in the composition of the first group are HP Quality Center, IBM Rational TestManager and SilkCentral Test Manager for Micro Focus. The group of functional and regression testing tools contains respectively HP QuickTest, IBM Rational Functional Tester and Silk Test. And as examples of the last group we can mention HP LoadRunner, IBM Rational Performance Tester and Silk Performer.

Beside them, some products provide capabilities for resource planning, risk and costs management (for example HP Quality Center, IBM Rational Quality Manager, SilkCentral Test Manager). In other words, first steps are made to provide solutions that cover more aspects of quality, including functionality to support test planning, requirements and defects management.

Leaders in the provision of integrated packages for test management and partly quality management (some to a bigger degree than others) are HP, IBM, Micro Focus, Oracle and Parasoft. Among the main features that most of them provide, we can highlight the following ones:

- event tracking and creating reports based on different metrics;
- some of the companies (like IBM) provide opportunities to reuse their already developed components;
- a unified environment for management of the testing process, integrated with functional and load testing tools;
- requirements management;
- collaboration work of remote teams and opportunities provided for online reviews of project development through performance indicators and statistics for all important events;
- planning tools based on risk assessment;
- partial capabilities to manage and prevent defects.

Let's look at the functions that the integrated web-based solution for lifecycle management of applications offered by HP—**HP Quality Center** provides.

According to the company it is a flexible, unified platform for management and automated building of reliable and high quality applications. HP Quality Center enables organization and planning of the various stages of an application development through systematic control over the processes. On one hand these are processes associated with managing software products requirements, and on the other hand, different types of testing, test management, version control and defects tracking. In other words, Quality Center covers most of the stages of the lifecycle of a software application (see Figure 1).

At each of these stages several functionalities are submitted:

- business analysts can define application requirements and related test technical specifications according to defined business priorities;
- quality assurance managers can prioritize planned efforts in the direction of testing, based on established business risk;
- possibilities for designing test plans and developing test scenarios, as well as their later implementation;
- ability to identify the defect and its sharing between developers in the organization, including the storage of information about its status, priority and release in which the defect was detected (defect tracking);
- business analysts and testers can control multiple versions of individual tests or components, while data integrity is also assured;
- project managers can review defined quality metrics and make decisions based on them about whether the current version of the product is ready to be put into operation.

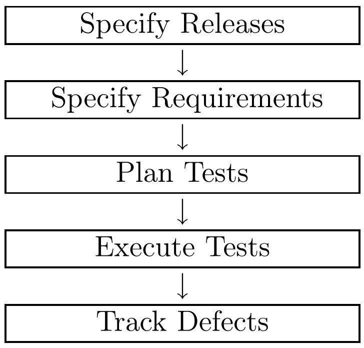

Fig. 1. Lifecycle stages of a software application development within the scope of HP Quality Center [8]

For its part, IBM Rational Quality Manager is a component of a scalable web-based platform through which Collaborative Lifecycle Management can be provided. In this integrated environment, IBM offers the use of individual products or several combinations of them, depending on the needs of the software organization. This in turn is a significant advantage because it allows greater flexibility and lower costs, in case we do not need certain functionality. Because of the open and scalable platform IBM Rational Jazz, additional tools such as IBM Rational Team Concert (teamwork management, versions control and maintenance of planned iterations), IBM Rational Requirements Composer (tool for project requirements analysis and requirements management) and a series of external tools can always be installed if necessary [2].

The main functionalities offered by Quality Manager are in the field of testing and very few of them overlap the requirements definition and their management. The most essential among them are:

- testing based on identified risks, which allows to set priority to tests depending on the risk degree and the extent of their impact;
- identifying duplicate defects in order to minimize the repetition in staff actions;

  planning the lifecycle of the testing process, defining the roles, processes and responsible testers, as well as the ability to create test scenarios and to link them to specific test plans;
- importing project requirements from external tools (for example IBM Rational Requirements Composer), storage of these requirements and their association with specific test scenarios or even a specific test plan, in order to allow tracking its coverage and execution;
- the ability to connect different environments (browsers, databases, operating systems, etc.) that are maintained and controlled within a defined test plan and use this relationship to generate test configurations and to track the execution of tests;
- ensuring interaction between remote teams by using an interface based on Web 2.0;
- the chronology of the methods and testing templates used with data for the specific version is stored in a central repository (data warehouse), and their reuse is ensured.

As a positive side of the IBM Requirements Composer tool, which can be used together and is tightly integrated with IBM Quality Manager, we can note the good possibilities for *requirements analysis*, which we consider an essential prerequisite for ensuring high software quality during its development. On the other hand the main criticisms against this instrument in relation to the requirements management are security, imperfect collaboration and limited reporting [9].

If we look at the scope of the **Silk** software package, which is owned by the company **Micro Focus**, we will observe again an insufficient coverage of all activities relevant to the software quality management. However, the company defines the tools in the group Silk as a complete solution for quality management, including an integrated testing suite to ensure that software is comprehensively tested and developed on the basis of the highest quality standards [10]. In this suite of solutions several products are included that provide different functionality as in the IBM suite mentioned above—SilkCentral Test Manager, SilkPerformer CloudBurst, SilkPerformer and SilkTest. SilkCentral Test Manager has the ambition to ensure quality from the beginning of a project by requirements and tests collaboration and provides continuous monitoring of quality actions and methods to achieve it. However, we think that the possibilities offered for planning and defining the processes for quality assurance as well as change control during software development are limited.

Based on the analysis presented of the reviewed tools' main features, we can summarize the extent of their implementation by the comparison shown in Table 1.

After analyzing the several popular tools and integrated test management and partly quality management suites, we can conclude that in fact they provide a solid base for building quality software applications and continue to evolve in this direction. However, it cannot be said that they provide a complete and comprehensive system of quality management. We will focus on key problems and barriers that should be solved and overcome.

In many existing applications for project and/or quality management, change management is only used by developers as the information is kept only for versions of individual fragments of code. To achieve higher productivity and control of the overall work on a project, we believe it is necessary to track and store history about other equally important components such as contracts, user requirements, quality plans, associated defects with specific version and risks.

In software development, the actions of the quality assurance team and testers are seldom automatically linked to the actions of developers. This is obvious in the abovementioned popular solutions for quality management and software testing. For example, a defect that is identified by a tester or by a

Table 1. Comparison of tools according to their functionalities / modules

Functionality/Module	SilkCentral Test Manager	IBM Quality Manager	HP Quality Center
Requirements Management	yes	yes, but an additional tool must be used	yes
Risk based testing	yes	yes	yes
Test plan	yes	yes	yes
Specification Management	no	no	no
Risk Based Quality Management	no	no	yes
Defect Management	no	yes, but an additional tool must be used	yes
Reuse of components	no	yes	there is no precise data on the presence of such a possibility
Control project against defined quality assurance metrics and procedures	no	there is no precise data on the presence of such a possibility	partly

person responsible for QA will be stored in a repository such as that offered by HP Quality Center, but it does not automatically connect to the development cycle which should work with it. On the other hand, when the developers release a new version of a program segment and send it to the quality assurance and testing departments, the status of each corrected defect cannot be synchronized automatically with the new code, i.e., it is not available for inspection, as there is no close coordination between the processes. This lack of integration in terms of tracking defects and their associated requirements cause unnecessary effort and a number of questions such as: what defects have been corrected and in which compilation (build); where they were registered and to which components of the software under development they are relevant.

Although the process of tracking defects is described in Fig. 1, after the analysis of the proposed application functionality was conducted, we found that it does not cover the automatic connection mentioned above or integration of defects controlling with requirement management processes.

Another significant disadvantage of the reviewed instruments is that none of them provide automatic and intelligent creation of tests, based on what the respective software is designed to execute. In most of the available tools scenarios are developed based on the actions that testers perform during the product testing, i.e., on the principle of "recording" and "reproduction" of test scenarios. In this case, the reliability and amount of coverage depend on how well and through how many different and possible scenarios the tester will go. Also, the results generated by the system must be evaluated by the tester to determine whether a test is passed or not, based on his understanding and knowledge of the specifications. In other words, none of the existing tools for automated software testing provide possibilities for the automatic generation of test scenarios. The main reason for this is the lack of means to present the specifications, requirements and design, so that they can be interpreted, understood and used by an intelligent tool for test generation.

All these issues require a more complex approach to the software quality management as well as the provision of a *suitable environment* for quality building and maintenance as an integral part of the entire product lifecycle. The undisputed fact is that the development of such a complex quality management system, covering all aspects, requires a serious software engineering team and years of work. This in turn requires the implementation of flexible architecture that allows necessary improvements to be made relatively easily and without significant modifications and investments.

3. Integrated system for software quality management. In the existing integrated quality management suites, testing and quality specialists are able to send information about any detected defect, but not to receive feedback on the results from Units tests and analysis of defects.

A system of software quality management requires the implementation of complex and diverse functions, including mapping data from different sources; making data consistent; transformation and upload into a repository of data (data warehouse); extraction of analytical information; implementation of a regulated reporting; maintenance of tools for execution of random queries; multidimensional analysis and many others. This in turn most often requires the use of different products, which leads to complication of the system architecture, because of the need for integration of heterogeneous instrumental environments, additional administration costs, problems with the coordination of data and metadata across multiple servers and/or applications.

At the same time, it is necessary for IT professionals to simplify and automate the process of business application development in the new hybrid world of virtual and cloud computing and to increase the benefits that modern technology brings to businesses, without taking unnecessary risks at the same time.

From everything said here, we can conclude that it is necessary to build an integrated architecture that allows process coordination and synchronization in the software organization in such a way as to allow different departments to work as one. Integration should be at both the consistency of activities among all participants and the level of IT systems used. In order to satisfy the need to manage the entire lifecycle of developed applications that provide possibilities for distribution of responsibilities, modularity, reusability, loose but also flexible coupling of various applications and resources as well as high scalability, we consider it appropriate to use **Service-oriented Architecture (SOA)**.

SOA is defined as a model for integrating business processes and maintenance of information technology infrastructure through secure, standardized components—services that can be reused and combined, in order to address changing business priorities [1]. The main part of the concept is that the different services communicate in a standardized way and can be accessed, without requiring knowledge of the technologies and platforms through which they are realized.

A significant advantage of the chosen architecture is its ability to achieve perfect integration of the new quality management system, which we submit should be built with the existing applications in the organization and achieve the desired level of integration of different modules. This can be done because the SOA architecture enables the connection of miscellaneous applications, systems and services in a heterogeneous environment which, frankly, any software organization usually presents. For these reasons it is the right choice for architecture to serve as a basis for developing a system for software quality management, which should cover all levels of an organization and also allow for a high flexibility and scalability.

In terms of modules and subsystems of the innovative solution we offer, its scope and the processes that it should serve can be represented most fully by its functional structure (see Figure 2).

The figure shows that the proposed quality management system will seek to cover all planning quality activities in an organization, as well as all activities that are necessary for meeting the goals of achieving quality through the management and control of the project development, test planning and management, requirements management, risk, defects and configuration management. The detailed description of each of these subsystems and their components is not covered by this article.

Developing such a system that integrates various processes such as quality planning, planning and project control, management of testing, risks, requirements, configuration and defects undoubtedly poses some risks. One of them

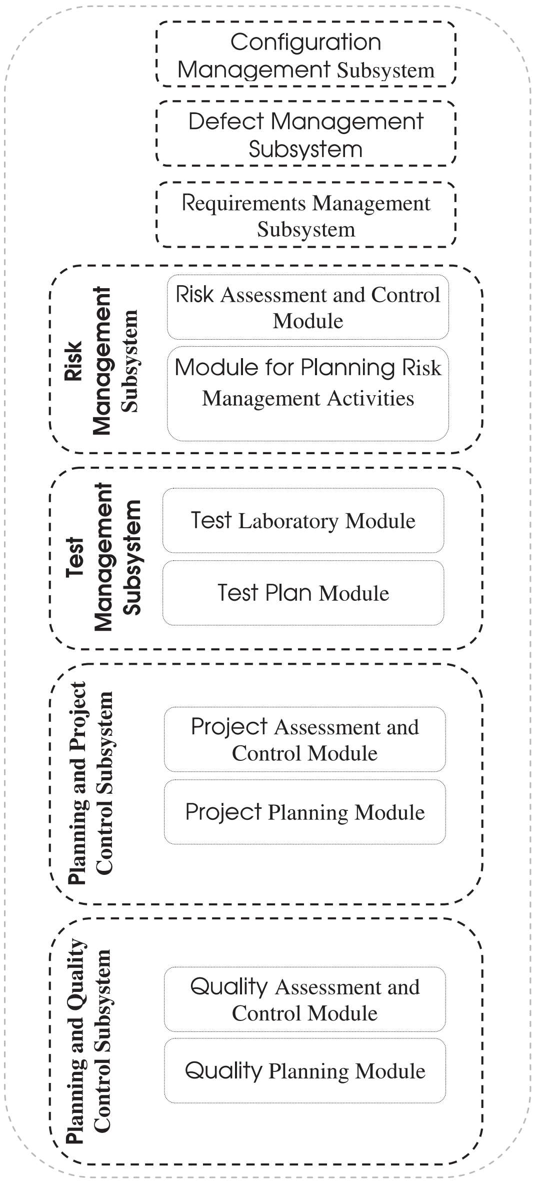
Configuration Management Subsystem
Defect Management Subsystem
Requirements Management Subsystem
Risk Management Subsystem
Risk Assessment and Control Module
Module for Planning Risk Management Activities
Test Management Subsystem
Test Laboratory Module
Test Plan Module
Planning and Project Control Subsystem
Project Assessment and Control Module
Project Planning Module
Planning and Quality Control Subsystem
Quality Assessment and Control Module
Quality Planning Module

Fig. 2. Functional structure of the software quality management system

is the complexity resulting from the need large data flows between systems to be transferred, transformed to a certain extent and synchronized. On the other hand such a decision will be linked to serious investments and the participation of highly qualified specialists, which would make it costly and unprofitable for the small and most medium-sized software firms.

It is known that for many of the listed processes companies use some software or automated tools. Their integrity, however, requires the consideration of many factors when choosing an appropriate integration strategy and also the availability of great technological competence and skills. Therefore, such an approach to integrated quality management would not be so suitable for small software companies because of its complexity, the requirement of specific skills and last but not least the requirement for established quality management practices and procedures. The latter condition is essential, because unfortunately software quality management is not strictly established and not laid down as a factor to be controlled during each of the processes listed above in small and even some medium-sized software companies. It can even be said that it is most often served only in the test phase. Thus we can conclude that implementing the proposed system may be too complicated and generally would not be of significant interest for such companies.

SOA architecture, as we emphasized, in turn provides excellent opportunities for integration of different subsystems, and it also saves a lot of costs in the long run. In other words, the initial investment of time and money necessary to develop a system based on SOA can be large, but the return is high [11]. As any approach, SOA imposes some limitations. The first is the complexity resulting from the fact that the transition to SOA is slow, requiring organizational and technological changes in the firms that have not yet developed this kind of applications. The second concerns the need to maintain high security of a system that connects several software components accessible from many participants over the Internet. Finally, the potential high cost of training, a highly skilled workforce and implementation of loose coupling that requires a large initial investment could make development significantly more costly in the short run.

Still the concept of building a unified and coordinated system of quality management that integrates all processes in the development lifecycle is highly topical. The findings in many studies support this, such as [3], according to which *the close collaboration between teams* will be of bigger and bigger importance to the quality management in an organization, *the possibility of coordination and synchronization of activities* so that different departments *function as a whole.* To reinforce the validity of the proposed concept of integration through SOA we can give as an example some authors' belief that the trends toward the realization of many innovative software solutions through SOA and the move to SOA can be a catalyst for change in the culture of Quality Management [5].

4. Software quality management system based on SOA architecture. Service-oriented architecture allows for separate logic units (services) to exist independently, but not isolated from each other. It is required that the services be able to satisfy certain principles which allow them to evolve independently, while maintaining significant community and standardization. One of the most commonly used systems for SOA solutions building are IBM WebSphere [15], TIBCO ActiveMatrix BusinessWorks [12], Microsoft BizTalk [4], Oracle SOA Suite [7].

In the presence of integration the links between the code and problems as well as between changes and the test plan module can be viewed and analyzed (see Figure 2). The result is much better planning and realistic deadlines.

The systems built on the service base (SOA) are characteristically composed of independent software objects whose functionality and metadata are accessible through a unified interface. Most often this access is realized using web services, which ensure the use of different functional components through the global Internet network. This type of systems represent a new generation of complex distributed systems that allow remote users (outside the organization) to interact with each other and to work with the same information in real time.

In the context of the proposed *software quality management system*, this unified access to data from multiple sources with different formats and degree of structuring would be extremely useful. This is especially true when many external developers are employed for a project (outsourcing) or even in order for the opportunity for direct communication with contractors and clients to be provided. Thereby each participant in the development of a software product would have uninterrupted access to the resources needed to perform the tasks in a place convenient to him/her and regardless of the device used—desktop PC, laptop or mobile device (smartphone, tablet, etc.). All this implies some difficulty in building such systems as well as special attention and consideration of all possible risks in terms of security. The complexity of the SOA systems is due to the new architectural approach to integration of program units that operate in different environments, in other words use and interaction between services regardless of the programming language and technology they were built upon [17].

The key aspects of SOA are services, components, service composition and choreography. An SOA-based system consists of different layers that are presented in Figure 3.

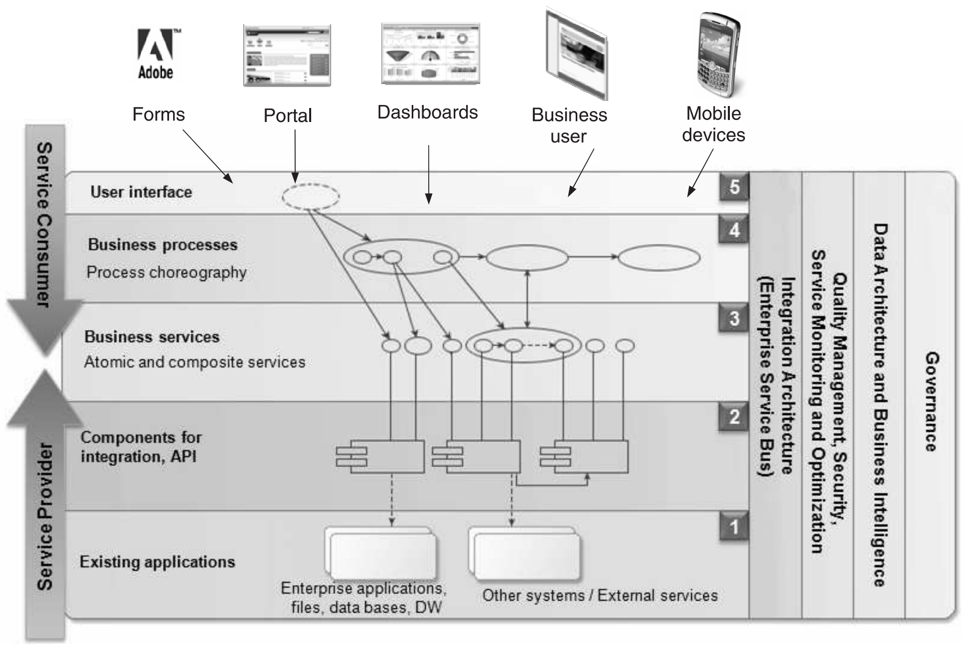
Adobe
Forms
Portal
Dashboards
Business user
Mobile devices
Service Consumer
Service Provider
User interface
Business processes
Process choreography
Business services
Atomic and composite services
Components for integration, API
Existing applications
Enterprise applications, files, data bases, DW
Other systems / External services
5
4
3
2
1
Integration Architecture (Enterprise Service Bus)
Quality Management, Security, Service Monitoring and Optimization
Data Architecture and Business Intelligence
Governance

Fig. 3. Extended architecture model of a SOA system based on [18]

The first layer includes the existing enterprise applications (both custom and packaged ones), databases and data warehouses, which despite their diversity can be accessed by users or service consumers in the form of shared services over the Internet. This layer forms all information assets that can be reused. The next layer (2) provides program units or components that provide specific functionality, but are not designed to solve business problems. Subsequently, these components can be connected or integrated in a certain order, so as to meet specific business needs.

Moreover, if the business changes a requirement, thanks to the loose coupling between the components in SOA, the necessary changes will be implemented much faster and easier.

Any decision to automate specific activities includes the definition of appropriate processes. The business process consists of logic that determines the actions that must be taken. Each task is encapsulated in a particular service, which represents a step in a process. A service can be atomic or composite—to cover the logic provided by other services. On the other hand, a process can be composed of several services (for example business transactions) and that process can be presented as a service.

One of the key principles in the SOA architecture is precisely the *loose coupling* of services, which we will discuss later. Typical of services is that they must interact with each other, although they are loosely coupled. In other words, a specific type of medium must be provided to allow communication between different services and applications. This is ensured by the so-called **Enterprise Service Bus—ESB**. Each service connects to a common bus through an adapter that transforms data to a common format. In this way data from different sources can be accessed even if they are supported in different systems, on different platforms or different DBMS (Oracle, DB2, SQL server, Informix, MySQL or PostgreSQL). The service bus provides many other functionalities such as intelligent routing of messages, data and message transformation, reliability and security control, service governance, monitoring and logging [14].

SOA allows both integration of applications that actually are providers of certain functions or operations (services) and business process integration across the entire enterprise. This means coordination between different departments within the organization and its partners, i.e. all stakeholders work in a unified environment. In the context of the idea of developing an integrated software quality management system, the SOA approach will allow close communication and a possibility for the coordination of activities among all units involved in the development of a particular application. To achieve effective management and control of the processes, a tool should be provided that can seamlessly communicate with the service bus. It should see all services related to it and allow their calling in a specific sequence. It shall thereby implement the *choreography of processes.*

Once processes are developed, a mechanism for their use must be provided. The presence of interfaces is needed in order to enable users to interact with business services and processes developed on their basis. The focus is on providing web-based access, as there will be many participants in a business process within the system that should be developed, so they will require a different access method. Also, the process implementation most often requires the use of several applications.

Having in mind all these considerations, when designing a system, it is necessary to provide portal interfaces that allow interaction between users through portals, information dashboards and various types of mobile devices. Services are characterized by attributes that define the service quality or policies on how interfaces can be used by their consumers. These interfaces are based on standard protocols (usually those of web services, but this is not the only possible implementation) [16].

On the other hand the communication protocols used for interaction be-

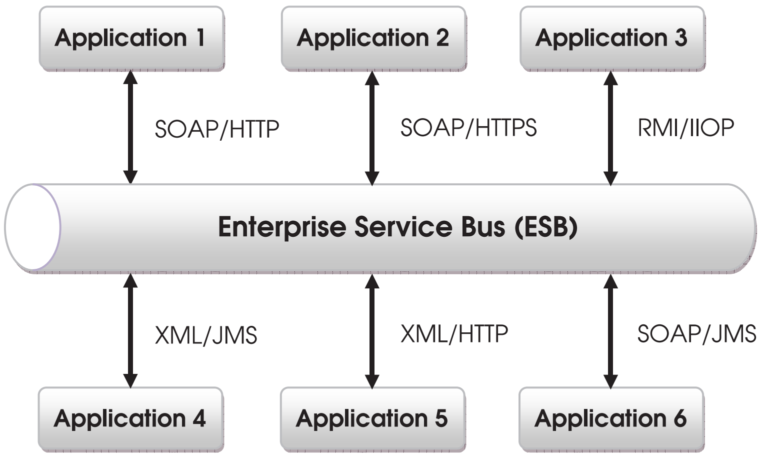

Fig. 4. Interaction through the Enterprise Service Bus

tween the services must be compatible with a wide range of platforms. Therefore protocols such as HTTP, HTTPS, JRMP, IIOP, and JMS are used here. Thanks to the enterprise service bus, individual applications are able to easily connect and interact with each other, even when their interfaces and protocols do not match completely (see Figure 4).

As already noted, it is very important that the individual modules and subsystems are able to communicate with each other. The service bus will play a major role in the exchange of messages between components in the system. In this way an integrated management of requirements is provided both between teams and between different phases of the project.

The enterprise service bus enables a two-way communication among all components in the processes of extraction, transformation, loading, integration and recording of information. It improves the sharing of services and their reusability while reducing the number, size and complexity of the interfaces used.

The main tasks performed by ESB can be summarized as follows:

- defining the service, which must accept the inquiry and its redirection to respond to the relevant request;
- transforming the transport protocols between the source of inquiries (requests) and related services;
- transforming the format of messages between the data source and data consumer;
- event management across the different data sources.

When we talk about services, it should be noted that in the basis of service-oriented architecture there are three main web standards—SOAP, WSDL and UDDI. SOAP is an XML-based protocol for defining the rules for exchange messages between the web services and is independent of the network, transport and the programming language. WSDL provides the basic information that identifies the service and enables its calling. In other words, it provides a standard way for interface specification and details about the service—operations, location, linking method (i.e., how the service is called—most commonly SOAP/HTTP and SOAP/JMS). UDDI maintains a central register of all existing services. Through it they can be searched, published or used for information. That is why through the use of such common standards as XML and HTTP, web services can access various Internet applications, in order to enable the realization of business to business (B2B) solutions, to integrate multiple applications inside and outside the organization.

The advantages of SOA architecture for the implementation of the concept of automated software quality management can be summarized in the following areas:

• **Integration which is not dependent on the programming languages used.** The wide distribution and application of web standards creates exceptional conditions for cooperation and integration that will continue to improve with the development of service-oriented architecture. The foundation of the modern standards that use the web services to communicate with each other is the XML language, which is designed in order to standardize, unify and simplify the storage and transfer methods of different kinds of data. In other words, it provides developers with an opportunity to model data regardless of the source, language and platform.

• **Reuse of components.** The possibility of developing a software component in the organization and its presentation in the form of service with possibility of reuse would help achieving much higher efficiency. With the right approach to the service design, such a service interface could be built that allows service application in more than one decision and so avoids the duplication of functions. Creating components that can be reused, such as algorithms, methods, processes, documentation requirements, tests and so on, helps improving the efficiency and quality of any software application development. When various reuse scenarios are planned, the services disclose their potential for building key components of the quality management system, which then can be shared between different teams in the organization as well as individual projects that have the same or similar components. Also, many components can be combined to provide better functionality, which in the terminology of SOA is called “orchestration”.

• **Organizational flexibility.** SOA allows for the building of independent functional blocks that can be accessed through standard Internet protocols. If necessary, they can be rebuilt, used together or integrated into an application easily and quickly enough. Their flexibility is determined by the ***loose coupling*** of various tools and resources and reflects the concept of minimal dependencies. This in turn means that the modifications have minimal effect and the systems can still operate even when some of the services are made of incompatible technologies, or have suffered damage.

• **Better transparency—better management.** Web services are more visible and therefore more manageable when compared to web applications. By better visibility we mean that the status of a service can be monitored and controlled at any time. To achieve this the realized transparency plays a key role in allocating services in SOA architecture by using registers of services, brokers of services or other mediators, located between providers and consumers of services [16].

It is not accidental that the focus of many software engineers moves from object-oriented programming to one that separates the interface technology from the technology of implementation. Service orientation is focused on integration by using different interfaces to achieve the same advantages that are the goals of object orientation, but at an organizational level.

5. Conclusion. Considering the facts stated above, we can conclude that the SOA architecture meets the requirements specified in the beginning, aimed at defining a new approach to building a comprehensive, integrated and scalable system, consolidating heterogeneous data and processes into a unified environment for software quality management.

It will provide flexibility for different applications and resources, and an ability to monitor, control and synchronize the work of all teams in a software organization. Flexibility is expressed in minimal dependences of software components and their reusal, which will be increasingly valued in the field of software engineering, because they help to reduce development time, to achieve higher productivity of developers and higher quality of the developed applications.

At the same time it should be noted that such an approach to software quality management at the whole development lifecycle is a complex process requiring a gradual transition from the software organizations which would have implemented it. The complexity is due to the fact that an overall change is needed from a technological, methodological and organizational perspective in terms of software quality management. At the same time the idea that was presented – to build the system as a service-oriented solution – requires the reporting of a number of potential problems. Among them we can highlight the following:

- how the complexity of the proposed system based on SOA could be reduced or limited;
- need to define requirements and instructions on how to build high quality components;
- assessment of the scope of software services poses risks;
- how to minimize technological risks;
- how to build the right strategy to phased move towards quality management as an integral part of the software development lifecycle.

However the opportunities that SOA provides and the trends to its rapid development in the future are a valid premise for assuming that it is a suitable basis for building a system for managing software quality.

REFERENCES

[1] Bieberstein N., S. Bose, M. Fiammante, K. Jones, R. Shah. Service-Oriented Architecture Compass: Business Value, Planning, and Enterprise Roadmap. Pearson Education, Upper Saddle River, 2006.

[2] Cho A., C. Pampino. Collaborative Lifecycle Management "New and Noteworthy" in 2011. `http://jazz.net/library/article/655`, 16.02.2012

[3] Development Testing: A New Era In Software Quality. Forrester Research, Inc., 2011, 11–12.

[4] Fuel Innovation, Boost Business Agility and Get higher Return on Investments. `http://www.microsoft.com/biztalk/en/us/soa.aspx`, 2012

[5] Kaufman M., R. Dorin. Adopting a Lifecycle Approach to Software Quality Management. Hurwitz & Associates, 2007.

[6] Murphy T. Magic Quadrant for Integrated Software Quality Suites. `http://www.gartner.com/technology/media-products/reprints/microfocus/vol4/article1/article1.html?loc=hpfeature2`, 2011

[7] Oracle SOA Suite 11g.
http://www.oracle.com/us/technologies/soa/soa-suite/index.html, 2012.

[8] QC testing process.
https://www.akshayiyer.com/testmngtoverview.asp, 2012

[9] Schwaber C., M. Gerush. The Forrester Wave$^{\text{TM}}$: Requirements Management, Q2 2008. Forrester Research, Inc., 2008.

[10] Silk Software Test Management, Test Automation and Performance Testing. http://www.microfocus.com/products/silk/index.aspx, 2012.

[11] SOA: The Benefits and Challenges of Shared, Reusable Services, http://www.thebrookfieldgroup.com/news_story40.php, 2012

[12] TIBCO ActiveMatrix BusinessWorks. http://www.tibco.com/products/soa/composite-applications/activematrix-businessworks/, 2012

[13] Visitacion M., M. Gualtieri. The Testing Tools Landscape: 2010. Forrester Research, Inc., 2010.

[14] Waseem R. SOA-Based Enterprise Integration: A Step-by-Step Guide to Services-Based Application Integration. McGraw-Hill, 2009.

[15] WebSphere software. http://www.ibm.com/software/websphere/, 2012

[16] Димитров В. Ориентирана към услуги архитектура. “ТехноЛогика”, 2009. (in Bulgarian)

[17] Илиева С., Вл. Лилов, И. Манова. Подходи и методи за реализация на софтуерни системи. “Св. Климент Охридски”, София, 2010. (in Bulgarian)

[18] Сервисно-ориентированная архитектура (СОА). Теория и практика интеграционных проектов. IBM Corporation, 2008. (in Russian)
http://www.gsom.spbu.ru/files/upload/career/presentations/IBM_28_04_08.ppt, 2012

Olga Marinova
University of Economics – Varna
77, Kniaz Boris I Blvd
9000 Varna, Bulgaria
e-mail: olga_tuleshkova@abv.bg

Received March 20, 2012
Final Accepted July 19, 2012