The necessity of integrating enterprise service bus (ESB) with proactive components (agents) was described in previous paper [6]. In this paper we show some practical results we achieved integrating our enterprise information system bus that is built on Oracle SOA platform [8] with agent system JADE [1].

I. INTRODUCTION

Many companies use web services within SOA [4] [5] [7] architecture to support their business. SOA theory promises IT-professionals many benefits stemming from SOA. Implementing new architecture in our telecom company we found many drawbacks that forced us to look for a way to improve SOA. One of the major problems for our design team was service management.

All our services are exposed to enterprise service bus that has domain structure. Standard tools that support bus management do not allow searching for specific service. With the number of services to grow it becomes a problem. We also would like to share our web service directory with our partners, analysts and other people without giving them access to management tools. We found that existing functionality of management software for ESB is very weak and had no choice to go for our own solution – integration with software agent platform.

Agents can work autonomously and be proactive in achieving their goals [15] [16] opposite to services that are passive unless they are invoked.

Since our SOA platform is built on JAVA technology it was natural to select JAVA-based agent framework. We found that JADE is very mature project and also has such feature as web service gateway.

II. WEB SERVICES AND SOFTWARE AGENTS

The first bridge between web-services and JAVA agents was made by Whitestein Technologies and their product WSIG [3] (Web-Service Integration Gateway). They found many points of contact between these two technologies and propose method how to integrate two worlds. Their model is shown on the Fig. 1.

Joining agents and services gives intelligence for web services that is missing for now.

III. MAKING AGENTS WORK

In the next Fig.2 we introduce our vision how agent’s platform can help us automate routine work. Many of the system parts are already up and running while the others are our potential to grow.

A. Transform ESB metadata into agent ontology

Oracle ESB agent performs translation function between server metadata and agent’s internal structure. It gets XML data from the server API, extracts and maps oracle objects to agent’s ontology concepts and caches this information for 15 min after which an update has to come. Agent provides this information to other agents upon the query. ESB agent implements 2 behaviours – one to update its knowledge base and other – to reply for services list request. ESB agent is core part of the system and its behaviours are cyclic opposite to applet agents that work only in time period from client page open to browser close.

---

1 Web-services, due to their passive nature, can’t discover services published by agents and dynamically use their functions.
B. **ESB service lists for development team**

Data extracted from Oracle ESB server is already interesting for the developers and customers. We created applets to display information in a convenient way. User can search, sort and filter output using different parameters and get extra information from WSDL <documentation> tags.

C. **WSIG proxy agent**

WSIG proxy agent translates SOAP message requests/responses into agent ACL. It also maps WS schema with agent service description. Agent allows calling agent functions as web-services and vice versa. If UDDI option is activated agent registers services in UDDI register.

D. **Agents as intelligent web services**

Every Oracle ESB service can be represented by agent that publishes service operations using WSIG proxy agent in WSDL [14] format. This conversion from service-to-service is needed to make service proactive and have more control over client’s connections. Oracle ESB service control is far from being easy and informative enough. Client’s statistics suffer from address and process data absence.

E. **Oracle WS call**

Agent calls web service function after client calls service representation at Agent ESB.

For the customer there is no difference between calling Oracle ESB directly and calling agent representation of the service at Agent ESB.

F. **Agent’s WS management**

Since agent is situated in the middle between customer and web service it can carry many useful functions like gathering client’s statistics, notifying client about service updates or managing subscriptions. These functions can be exposed as applets on application server. WSIG supports automatic SOAP envelope request/response generation and WS invocation through server applet. This is very useful for testing purposes when client needs to know if service is alive without running special software like soapUI [10]. In fact this can also be automated by agents. They can call web service and check if the reply is reasonable (i.e. has no error in response) and set appropriate status for the service. The problem here is services that perform insert/update operations in databases.

Furthermore agents can store statistics into the database to be less dependent on platform failures and amount of system memory (not shown on the picture).

G. **UDDI service**

UDDI is widely used standard [11]. WSIG has ability to publish service in UDDI registry. Here, again, agents can automate this manual work.

For the described configuration of the system we recommend using either jUDDI [17] or UDDI4j [12] (default for WSIG) software.

H. **Open agent platform**

JADE is an open agent platform that follows FIPA [2] standards and can adopt many different types of intelligent agents. JADE can also work in a cluster with other JADE platforms. We don’t use this ability like we don’t use UDDI at our work, but we can use it in the future to load balance between client and server.

IV. **CONCLUSION AND FUTHER WORK**

We used agents to improve our everyday work with commercial off-the-shelf product – Oracle SOA suite and particularly in its weakest part – Oracle ESB.

Implementation of the system gave us a number of benefits compare to the original system [8]:

1. Lower down response time to the client
2. Ability to search services by name, domain, deployment time, etc.
3. Ability to read service and operation description (documentation) directly from WSDL
4. Ability to share information about services availability between different user groups (developers, partners, customers, analysts, etc.) without getting access to the system
5. Ability to connect to the RPC-style services
6. Possibility to create “active” services like agents

---

2 Publishing operations are manual. See example here [9]
The next step is to improve single services and bind them with WSIG. This step requires database for statistics and applets for management. As the primary goal was to prove concept of successful integration of web services with agents, future work will either be outsourced to professional programmers or be marked as internal company standard and developed inside.

ACKNOWLEDGMENT

Best regards for support to Estonian telecom operator “Elion” where all this work was done.

This work was partially financed by Estonian ministry of education grant SF0140113As08 under the title “Proactivity and situation-awareness”.

REFERENCES