Model of Data Exchange between Heterogeneous Systems Using Mule

Nenad Milanovic, Tanja Milicevic
University Computing Centre (Srce), Zagreb
Marohniceva 5, HR-10000 Zagreb, Croatia
nenad.milanovic@srce.hr, tanja.milicevic@srce.hr

Abstract. In today’s world of advanced technologies many systems are created on different platforms, languages etc. Integration of such systems can be difficult and time consuming.

Mule technology, based on Enterprise Integration Patterns, makes integration easy to implement.

Keywords. System integration, Mule, Enterprise Service Bus (ESB), Enterprise Integration Patterns

1. Introduction

Transferring data from one or more sources to destination of our choice is what data exchange is all about. The source can be planned and build in such a way that it is easy to integrate one way transfer into central part. But in practice it is not a case. Usually, data destination is completely different considering dedication, data type, even with OS and DBMS. Sometimes, more than few such systems must be integrated in secure, reliable and error corrective way.

Our goal is that two or more these kinds of systems work together as one big system. It is important that all systems are stable and independent so they can work regardless of each other. Systems that are integrated are mainly created on different platforms, using different languages etc. what makes them difficult to implement and therefore we need integration framework that can be easily implement. One method of integration is called ESB (Enterprise service bus). In ESB, a specialized subsystem (BUS) is added to the system in order to communicate with other subsystems. ESB provides an abstraction layer on top of an implementation of an enterprise messaging system. Enterprise message model defines a standard set of messages that ESB will transmit as well as receive. There are several ESB implementations available: Mule [5], ServiceMix, OpenESB and few others. Mule was chosen because its simple architecture allows easy implementation.

Using Mule technology we can generate packages by using data chunks and series. This new formed data chunks wait to be transferred to the other system. If any error occurs during the reception same packet can be resend. Setting Mule as middle layer, systems aren’t overloaded and can have their “own life” with little or no changes at all. The whole process, like preparing data, concerns about delivery and some similar actions are often at different place and generally on different hardware.

2. Background

Information System for Higher Education (ISVU) [4] is an information system made for institutions of higher education in the Republic of Croatia. It is financed by the Ministry of Science, Education and Sports (MSES). The implementation covers a complete evidence of syllabuses, the educational program fulfillment, course scheduling and all the students' activities, starting from the entrance examination, education and examinations to the graduation thesis and getting graduation certificate (diploma) and diploma supplement. Great number of users uses this system daily.

ISVU is network oriented modular system for data processing and higher education institution’s (HEI’s) services and departments intercommunication. It consists of several modules, some web modules and some desktop modules. The system is based on relational database management system (DBMS) - Informix Dynamic Server (IDS).

Students’ Nourishment Information System (ISSP) [6] has MS SQL server as DBMS and it is quite different from ISVU considering data organization level. ISSP is mainly card-financial management system which main purpose is to provide correct subvention for funding student’s nourishment. Since the government provides this it wants to control spending and takes logging of every food consumption authorized by student card.
There are several levels of funding, depending on a great number of factors such as distance from home, ECTS point collected during previous year, sport results, etc. Each of those records can be found in ISVU. For universities that are in ISVU data is collected and merged with ISSP data. Although there was a possibility to collect whole data through ISSP system, it was decided that it is better for student office to have only one application instead of two. Process of student’s evidence is faster with minor mistakes and we were able to achieve higher level of user satisfaction. Therefore accommodation data transfer had to be build.

3. Mule framework

Mule is a lightweight Java-based messaging framework that allows you to quickly and easily connect your applications and enable them to exchange data. Mule uses service-oriented architecture (SOA), enabling easy integration of your existing systems. Mule is able to connect systems regardless of the different technologies the applications use, including JMS, Web Services, JDBC, HTTP, etc.

Mule is based on ideas from Enterprise Service Bus (ESB) architectures. ESB allows different applications to communicate with each other by acting as a transit system for carrying data between applications using intranet or Internet. There are several types of ESB implementations. One difference between Mule and a traditional ESB is that Mule only converts data as needed. Typically in ESB you have to create an adapter for every application you connect to the bus and convert the application’s data into a single common messaging format. In Mule, the information is sent on any communication channel, such as HTTP or JMS, and is translated only as needed along the way. The advantages of using Mule are:

- Mule components can be any type you want. You can easily integrate anything from a "plain old Java object" (POJO) to a component from another framework.
- Mule and the ESB model enable significant component reuse. Unlike other frameworks, Mule allows you to use your existing components without any changes. Components do not require any Mule-specific code to run in Mule, with no programmatic API required. The business logic is kept completely separate from the messaging logic.
- Messages can be written/shown in any format from SOAP, XML messaging, WSDL service contracts, etc.
- Mule can be deployed on a different of topologies, not just ESB.
The Mule architecture and terminology use the principles described in the book “Enterprise Integration Patterns” by Gregor Hohpe and Bobby Woolf [3].

Our systems need to function independently. Processing of data exchanged shouldn’t affect applications work. Mule solves this problem by providing a messaging framework that reads, transforms, and sends data as messages between applications. A message in a Mule is data packet that can be handled and sent between applications using specific channel (also called a queue). Fig.1 describes a simple framework where two applications exchanging data using Mule. Both applications are connected to Mule. Mule reads the data from the first application, transforms it if necessary and then sends it to the second application.

![Figure 1. Mule messaging framework](image)

4. Implementation

There were two way approaches that could resolve transfer project. Two different teams had resolved implementation by solution in which no system suffered down-time or users delay. One team was closer to ISVU and another team worked on ISSP. Both systems are used daily by a great number of users so it was not possible to stop them for a few days to test in production.

Basic idea was to take the most of present table’s not increasing data in databases. There is log table in every enterprise information system so, by monitoring sequential ID of such tables, it is possible to keep track of changes in every moment and transfer only relevant data to the other system. At ISVU side we built pull solution which transfer date toward Mule in given discrete time frame. Mule connects to tables and collects data that are required. Afterward Mule makes data package which will be delivered to ISSP system over web server. Although packets
could be in any format we have chosen XML because of simplest integration on web server after receptions.

4.1. Implementation in ISVU

Mule manages communications between distributed client service components, named as Universal Messaging Objects (UMOs). A UMO is a plain old Java object (POJO) that receives and processes messages, and then communicates with the rest of the Mule-managed service components via inbound and outbound "messaging endpoints". These messaging endpoints are transport configuration components that have such characteristics as transport protocol and communication address. An endpoint is a specific channel on which a service can send messages and from which another service can receive messages. Endpoints can be divided in three groups: inbound, outbound and global endpoints. An inbound endpoint receives messages via the associated transport while outbound endpoint sends messages via the associated transport. A global endpoint acts as a template that can be used to construct an inbound or outbound endpoint elsewhere in the configuration by referencing the global endpoint name. Using the inbound and outbound routers we attach the UMOs to endpoints. The inbound and outbound messages are transformed by an appropriate transformer attached to the UMO. We also intercept messages for logging. Fig. 3 illustrates our model. Quartz, an open source job scheduling system, was used to trigger the events in Mule. When triggered, Mule gathers the data, acquired by ISSP, from ISVU database. After that data are transformed into XML. This transformed data are send to the web services using JMS (Java Messaging Service).

4.2. Implementation in ISSP

On the ISSP side of Mule transfer there is Microsoft technology. There is IIS web server which provides main place for file transfer and place for transfer initialization by triggering a simple application function. Firewall provides security to this server which is made for this purpose only. Mule sends request to function, web server takes XML packet and starts SQL server Business Intelligence module for transforming and writing records in adequate tables. After the whole process web server sends message to Mule about successfully finished transaction.

Records must be transformed before they are written in tables at destination side. Considering that there are no two same record type or logic, SQL Server Business Intelligence does transformation and error detection. Several error levels are established and maintained. Only administrator is noticed if some error levels are below predetermined critical level. He has to deal with the problem with smaller intervention over the record. Bigger errors, like not preserving natural order of packets or missing piece of information sends signal to Mule which tries to recollect information. One of Mule’s important task is logging all operations regardless if it is success or a failure. The ISSP model is shown in Fig. 2.

![Figure 2. ISSP model](image)

5. Conclusion and Future work

One of the main demands during integration of two systems or utilization of data from one system to the other is non-obstructing into activities of the existing system; into change of procedures, generally in non-obstructing the functionality of systems. Therefore monitoring, administration and maintenance of the system is easier to ensure, with the same consumption of working hours as before the intervention. In this article, one of the possible resolving methods for such problem was described.

Using this model, system is significantly easier to control. It is also possible to optimize server time if there is a problem with the server over-load whereby better activity of the whole system can be archived. It is possible to resolve the old technology problem if the source system upgrade is not an option while analytic services are still necessary. With a model that is easy to understand and easily procurable, exchanging
anomalies can easily be spotted. Built in detection of malfunctions can result in self-correction if some of the mistakes are predicted in advance. It is easier to monitor the activity of two systems with alarming working personnel in due time because third server is included in process.

We have resolved all mentioned issues and now both systems are successfully exchanging data leaving the largest task to Mule. There was data adjustment in both exchanging directions achieved so there is no need to create new table layouts or the conversational procedures.

Layer for data transfer is loaded in the most regular way with HTTP protocol and XML files, which is a standard today so this also does not demand complicated administration, and all existing hardware structures without further financial investments in software and hardware can be used. Processing and modifications itself are very simple once the system begins to function so development savings can be achieved. The Mule technology is simple and cheap solution for many complex problems considering sharing data between systems.

6. References


[6] Students’ Nourishment Information System (ISSP), https://www.cap.srce.hr/ [01/12/2008]