COM Interface to Rating Estimation System

Vladimir Nikolov, Yanka Yanakieva, Plamen Paskalev

Abstract: *The paper considers the integration of Portfolio Management System to remote Rating Estimation System using Web Services. The legacy System is a C++ application with CLIPS GUI designed to map the objects of the B2B Interface. The proposed approach includes two COM interfaces that transfer requests and responses from/to the Portfolio Management System to/from the Rating Estimation System.*

Key words: *System Integration, Intelligent Interfaces, Rating Systems, Rating Estimation Models, Computer Systems and Technologies*

INTRODUCTION

Web Services are established as a technology for system integration between companies over the Internet. They offer infrastructure for a secure Point-to-Point integration whereas the communication happens over the Hypertext Transfer Protocol (HTTP) using the Simple Object Access Protocol (SOAP) for the remote function call. SOAP applies the Extensible Markup Language (XML) and is a W3C standard of data exchange between applications. The using of XML makes Web Services language and platform independent and allows remote access to their functionality. The communication security is ensured by the Secure Socket Layer (SSL). Web Services are integral component of modern IDE's (for Example Sun Java2 Enterprise Edition (J2EE) or Microsoft .NET) which give the programmer tools for easy development of client applications consuming Web Services.

THE PROBLEM

The Rating Estimation System is developed with Java 2 Enterprise Edition (J2EE) as a data processing system estimating the credit rating of clients according to the Basle II requirements. The system uses a flexible scoring and rating calculation mechanism based on balance data and soft facts analysis. Industry branch influences form specific criteria used in the rating estimation. There is a possibility to correct the calculated ratings by credit officers of the client bank. Analyses of the input data are reported in a Basle II conform reports. The Rating Estimation System works at the same time as a data accumulation system saving data in a common data pool (Enterprise Storage Server (ESS). A statistical analysis of the stored data would result in future correction of the rating calibration formulae.

The Rating Estimation System is supplied with various integration products: B2B, RatingFactory, RatingFactory Online, Online Client (see Figure 1 below). The **B2B** interface (offered over WSDL and SOAP) is designed to access the whole functionality of the Rating Estimation System within variety of Use Cases. The **RatingFactory** represents an abstract interface supporting mainly the Use Case "Rate a client". The complexity of the rating procedure is reduced by minimizing steps and the B2B calls. The Java interface is transformed in HTTP interface by the **RatingFactory Online.** Input data are prepared and transferred in the FfiTs format (Flat File Transfer). The **Online Client** is a HTML Interface alowing easy data input and control of the rating procedure. The great disadvantage of the Rating Estimation System and their supporting products is the missing of an interface between existing Core Banking or Credit Management systems (called here **Legacy Systems**) and their Data Bases where amount of partner data already exists. So the integration to the Rating Estimation Sytem becomes a complex procedure requiring specific data mapping.

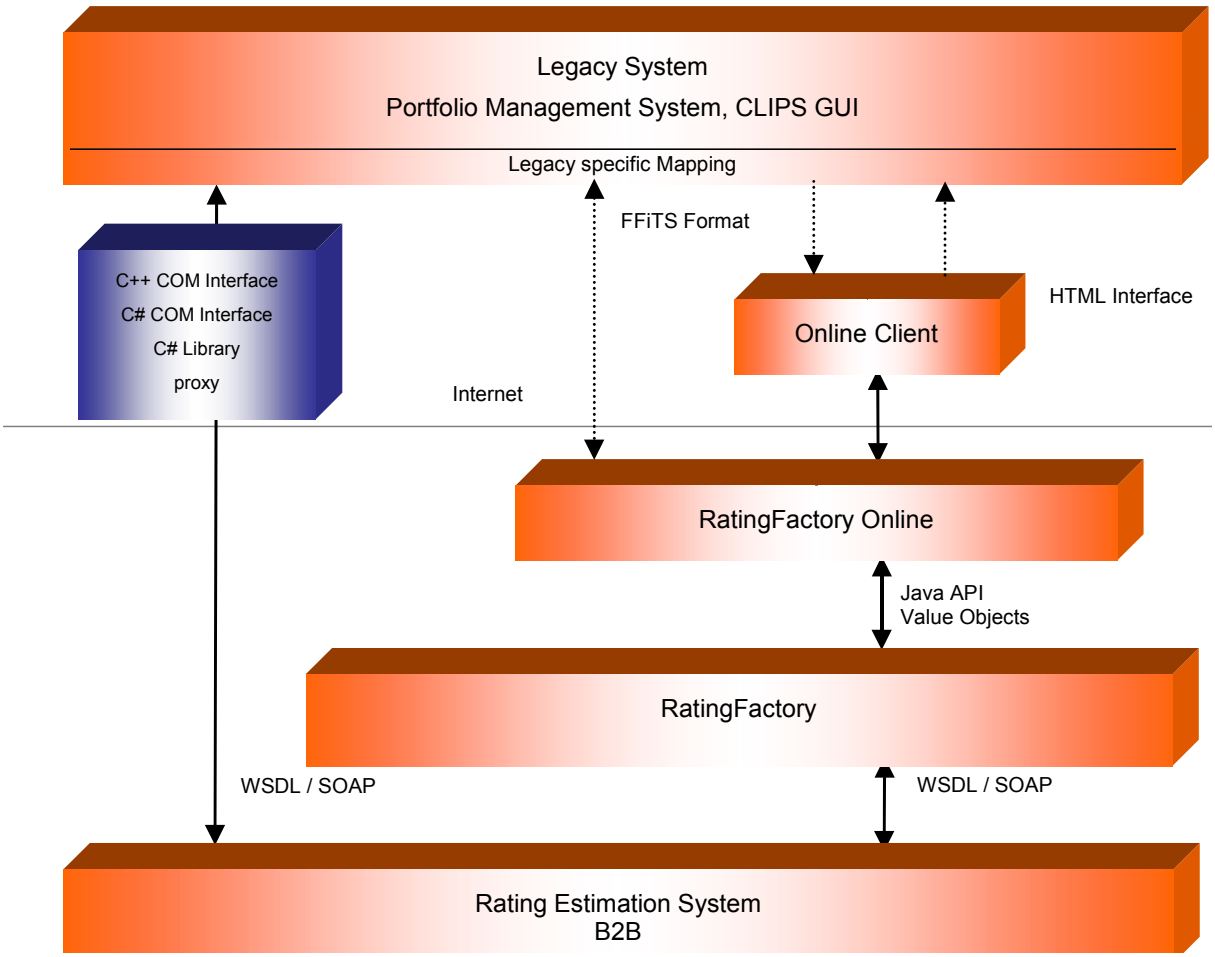

Figure 1: Rating Estimation System with integration products

The paper considers one approach for integration of legacy Portfolio Management System to the remote Rating Estimation System without support of the supplied integration products. The solution encloses direct access to the B2B from the CLIPS GUI over C++ COM Server, C# COM Server and the C# Library.

CONCEPTUAL SOLUTION

1. CLIPS GUI

The communication between the user and the Portfolio Management System is performed by the CLIPS GUI using a variety of script-based CLIPS models. The model scripts are independent from counterpart data and can be common for a large set of counterparts having similar behavior. The models are intended to input data, to load counterpart data from the existing database, to control the rating procedure calling interface functions and to receive the output from the Web Services (s. Figure 2). They correspond to the requirements of the Basle II consultative papers [3, 5] and conform to the data transfer objects of the B2B. For each transfer object needed in the rating process within the Rating Estimation System (WSKundeTO, WSBilanzdatenTO, WSSoftfactsTO, WSRatingTO [1]) is necessary to develop a rule-based CLIPS model including description of the graphical interface elements and rules for data verification and for function calls to exchange data with the B2B interface.

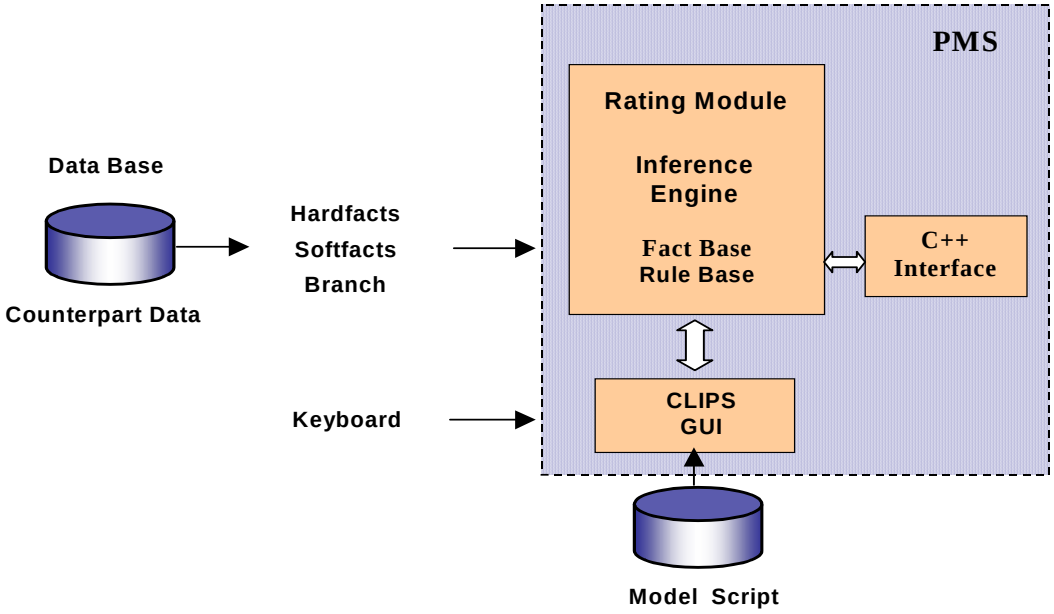

Figure 2: Portfolio Management System with the CLIPS GUI

2. IUI AGENT

In the described system some aspects of intelligent user interfaces were implemented. An intelligent interface agent was created, which is responsible for controlling the communications and intelligence aspects of the interface and is composed of three layers: Education layer, Implementation layer and Control layer. All layers consist of CLIPS rules groups.

The aim of the Education layer is to analyse user interactions and calculate statistical parameters.

The Implementation layer performs generation of rules, which modify the user interface in real time using the collected facts for the current user behaviour.

The Control layer is responsible for controlling the whole process as well as for correct selection of the users.

All control parameters can be changed dynamically, in real time, using knowledge base rules. The rule part of the knowledge base can be used in run time to change user interface, dynamically creating suitable interfaces for different users. Rules based on the user interaction (way of using the interface) can be created. The statistics of interface interactions can be analised and thus only the necessary information could be displayed. Based on the knowledge base, the agent is able to suggest and perform some tasks, which it recognizes in the user behaviour (usually the most repeatable actions).

Thus, the Intelligent User Interface Agent fulfills the following main characteristics as they were defined by their authors:

- **Non-linear solution** The task that has to be accomplished with help from the agent does not have an algorithmically derived solution, otherwise it will be enough to use a "wizard" instead of an intelligent user interface agent. [8]
- **Adaptability** One of the traits of intelligent behaviour for an agent is the ability to adapt to its user's skill level, personality, or behaviour. [6].
- **Interface sharing** In order to give the user suggestions for tasks to be performed, the UI agent has to share the interface with the application it is guiding the user through. [7]

The user interface is fully accessible, i.e *scriptable* [7] for the agent. Thus, the agent module is able to complete the tasks, identified as sequences, the user is usually performing.

3. C++ and C# SERVERS

The C++ Interface functions take the GUI control values stored in the Fact Base to produce a XML string where the XML tags are the control IDs and the XML values – the control values.

In the Portfolio Management System, communication with .NET is supported by COM Interop Services (CIS) [9]. COM entry point information is described to be able to call .NET services included in a managed application. The main point of the interoperability between different platform is adapting C++ types in order to make them compatible with the .NET types.

All applications that communicate through COM interface could be clients for those servers. Examples of such are all Microsoft office products. With the help of a script language, all the parameters of the server can be prepared and its functions executed. XML format is used to make the transition of parameters and results universal. Every server has different set of input data. Some servers require varying number of parameters. The returned results also have different size in accordance with the supplied parameters and that is why XML format is used for the transfer.

The advantage of the chosen method is in including of semantical information for the transferred data. Besides the values transferred, the XML contains additional information for their type and standard format of data representation. The standard specification determines how to interpret the data sent and the received results. For instance, date representations in different systems that are parts of the global network could be quite different. All modules participating in the data transfer should recognize them. One of the most commonly used standards for unification of date representations is ISO8601tz. It ensures exact definition of the date elements like day, moth, year, time and Greenwich time hourly differences.

Example: XML description of a service for initialization:

```
<?xml version="1.0"?>
<ROOT>
  <OpenBV_Session xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <benutzer dt:dt="string">NetUser</benutzer>
  <mandantennummer dt:dt="string">7004</mandantennummer>
  …
  <kundennummer dt:dt="string">102050</kundennummer>
  <datetime dt:dt="dateTime.iso8601tz">2005-04-12T04:34:46.000000</datetime>
  </OpenBV_Session>
</ROOT>
```

The example describes a call to the OpenBV_Session service. The supplied parameters determine the user that begins the communication. They include the parameter name, for instance "benutzer", the type of the parameter data – string, and value of the parameter – "NetUser". The "datetime" example is describing a date type in the standard ISO8601tz

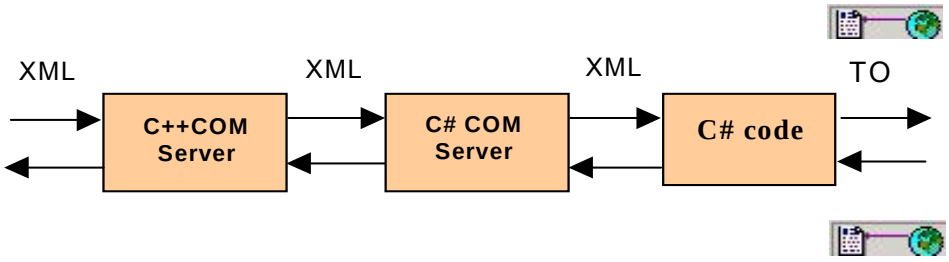

Figure3: XML based data Exchange

Figure 3 shows the modules participating in the services' calls. C++ COM server prepares the input parameters of the service to be called. The name of the service and its parameters are included in a XML description and transferred to the C# server. Both C++ and C# servers exchange data through common memory space, allocated in the client side of the application. The C++ server writes the input XML string and the C# server – the result XML string in the common memory.

The C# server implements a COM server written in C# which is used to start the services. The service name and the parameters are extracted from the received XML. Afterwards, the object for data transfer with the service is prepared and the result values are received. The received results are stored in a XML string and sent back to the C++ server.

4. C# LIBRARY

The modules of the C# Library are DLLs developed in the .NET environment. They act for the RatingFactory consuming Web Services over the proxy server. The DLLs receive values from the XML string passed by the C# COM Server, store them into the fields of the request transfer objects (TOs) and call Web Services methods. The deserialized response TOs are transferred back to the C# COM Server. The rating procedure is minimized to 12 steps by grouping of B2B calls.

IMPLEMENTATION

A part of the CLIPS model with counterpart data from the database and from the keyboard input is presented below.

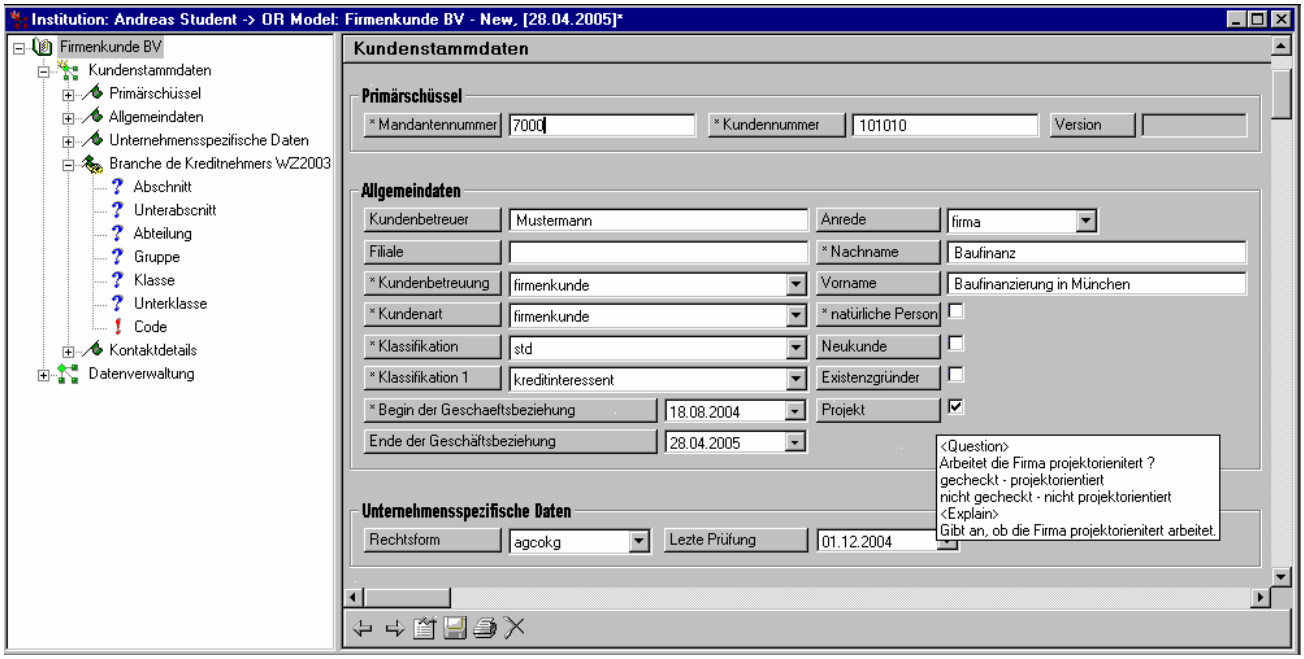

Figure 4: The CLIPS model of WSKundeTO

All values are stored in the fact base from where they are transferred to the C++ COM Server using XML string. Following are some parts from the XML string according to figure 4 interface inputs:

```
<benutzer dt:dt=”string”>User1</benutzer>
<mandantennummer dt:dt=”string”>7000</mandantennummer>
<kundennummer dt:dt="string">101010</ kundennummer>
……
```

CONCLUSIONS AND FUTURE WORK

The proposed approach for integrating the Portfolio Management System to the remote Rating Estimation System over Internet using 2 interfaces is verified with practical implementations. The following conclusions could be made:

- Counterpart data are exchanged directly between the legacy Portfolio Management System (C++) and the Rating Estimation System (Java) over Internet.
- The rating procedure is minimized by summarizing B2B calls in the C# library
- The C++ Interface and the C# Interface can be integrated to another C++ legacy System.

Future development concerns experiments and practical implementations in following directions:

- Deployment of different CLIPS models to cover the whole functionality of the Rating Estimation System
- Enhancing the properties of the Implementation and Control Layer of the IUI Agent

REFERENCES

[1] Spezifikation der B2B Schnittstelle 3.0a V1, 2004, BankVerlag

[2] A.Antonov, V. Nikolov, Y. Yanakieva, P. Paskalev "Platform Independent Rule-based User Interface",Computer science and technologis, TU-Varna 2004.

[3] A.Antonov, Y. Yanakieva, S. Petrova "Internal Firm Rating Model, International Conference on Computer Systems and Technologies - CompSysTech'2004

[4] V. Nikolov, A. Antonov "Rating calculation COM Server", FIRST INTERNATIONAL CONGRESS ON MECHANICAL AND ELECTRICAL ENGINEERING AND TECHNOLOGY MEET'2002

[5] Basel Committee on Banking Supervision, "The New Basel Capital accord", Consultative Document, 31July 3003

[6] Mandel, Theo, "The Elements of User Interface Design", Wiley, 1997

[7] Lieberman, Henry, "Integrating User Interface Agents with Conventional Applications", in Proceedings of the ACM Conference on Intelligent User Interfaces, San Francisco, 1998

[8] Dryer, D. Christopher, "Wizards, guides, and beyond: rational and empirical methods for selecting optimal intelligent user interface agents", in Proceedings of the 1997 International Conference on Intelligent user interfaces

[9] Esposito, Dino, "Building WEB Solutions with ASP.NET and ADO.NET", Microsoft Press, 2002

ABOUT THE AUTORS

Vladimir Nikolov Nikolov, Ph.D., Department of Computer Systems and Technologies, Technical University of Varna,

E-mail: nikolov@eurorisksystems.com

Yanka Nedelcheva Yanakieva, Department of Computer Systems and Technologies, Technical University of Varna,

E-mail: yanakieva@eurorisksystems.com

Plamen Paskalev, Doctoral Student at Department of Computer Systems and Technologies, Technical University of Varna, Project Manager in EuroRisk Systems Ltd.,

E-mail: ppaskalev@eurorisksystems.com