Intelligent Client of Rating Estimation Web Services

Vladimir Nikolov, Yanka Yanakieva, Plamen Paskalev

Eurorisk Systems Ltd.
31, General Kiselov Str., 9002 Varna, Bulgaria
E-mail: nikolov at eurorisksystems dot com
ppaskalev at eurorisksystems dot com
y_yanakieva at eurorisksystems dot com

Abstract: *The paper considers an approach for building of intelligent client of Web services compatible with a legacy C++ application. The program implementation is based on C++ COM Server and C# Library that transfer requests and responses from/to legacy system to/from the Java Web Services. The CLIPS GUI is designed to map the objects of the B2B and includes an intelligent agent to adjust the user interface in real time in dependence of the user behavior.*

Key words: *Intelligent Interfaces, Rating Estimation Models, System Integration*

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.

CLIENT STRUCTURE

1. Client tasks and structure

Programmers use mostly Sun Java2 Enterprise Edition (J2EE) and Microsoft .NET as development environment for the building of clients of Web Services because of the variety of programming tools they offer. In some cases the client of the Web Services have to be part of a legacy system (written in different language) which includes data entry module and supports partner data and credit risk data. The architecture of such client becomes more complex because of the different tasks to be performed. The client has to:

- offer an intelligent user interface for the input data
- load counterpart data from the legacy database
- map the input data with the input object of the web services
- ensure compatibility of the transferred data types

- call the web services methods
- receive results such as calculation values and error messages
- view the received results
- save the calculated results on the legacy database

The communication between the user and the legacy 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 (s. Figure 1). They 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. The models are flexible and correspond to the requirements of the Basle II consultative papers [3, 5].

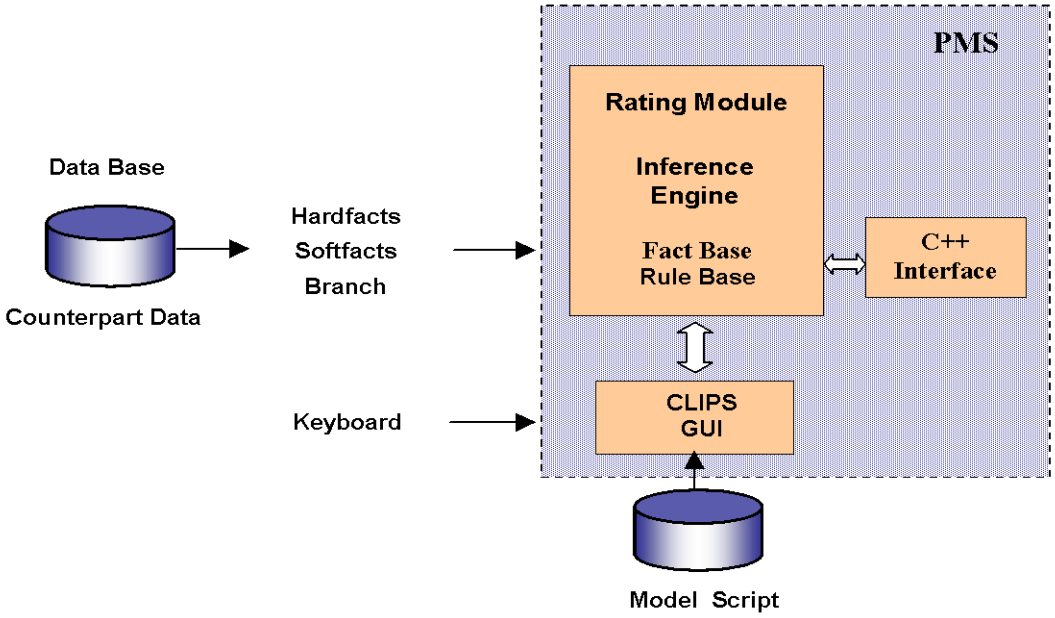

Figure 1 CLIPS Interfaces

The control of the rating procedure is performed by a separated model, which allows for calling the interface functions. They have to construct the requests, to receive the responses from the web services, to interpret the results and to show them finally on the CLIPS GUI (s. Figure 2).

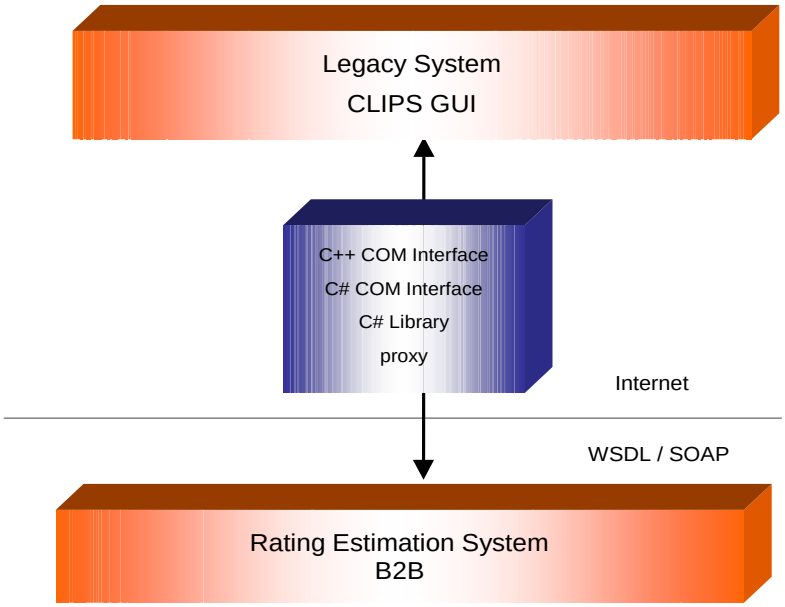

Figure 2 Multi layer client to the rating estimation web services

The problem with the compatibility arises because of the type differences in the object fields of the legacy system and the web services. To ensure compatibility data types are also transferred from the CLIPS GUI to the web services.

2. IUI Agent

The application is managing a lot of information. Sometimes there is too much information to display to the user. Techniques for determining what information is the most pertinent for the particular user are required, so the user is not overwhelmed with data.

Many researches in the area of Intelligent User Interfaces use rule-based intelligence ([10], [11]). Rule-based realizations could be divided in two main areas – representing uncertainty and dynamic user modeling. The addition and deleting of rules to dynamically model the user is an approach to determining answer reliability. Therefore, knowledge representations that can dynamically capture and model the uncertainty in the human-computer interaction can improve the modeling of the user and user interface states in an Intelligent User Interface (IUI).

In the described system some aspects of IUI 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 analyze 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 behavior. 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 analyzed and thus only the

necessary information is displayed. Based on the knowledge base, the agent is able to suggest and perform some tasks, which it recognizes in the user behavior (usually the most repeatable actions).

Thus, the Intelligent User Interface Agent fulfills the following main characteristics defined by the 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 behavior for an agent is the ability to adapt to its user's skill level, personality, or behavior. [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.

In the same way, as it could be done for the physical actions, communication acts can be formalized as plans. Communication acts are represented as operators in the plan library of a hierarchical planner [12]. Each plan operator defines the constraints and preconditions that must become true before the act applies, the effects (or post conditions) as long as the way of decomposing the act into a set of sub acts. Preconditions and constraints encode conditions (both physical and knowledge-based). The decomposition of a plan operator defines how higher level acts (describing an object) could be divided into potentially coordinated lower level actions. The approach is based on the assumption that not only the generation of text, but also the generation of the interface screen can be considered as a goal-directed activity [13]. We presume that there is at least one main act. The acts, supporting the main act are called subsidiary act [14]. As far as the acts can consist of main and subsidiary acts them selves, a hierarchical structure could be created, which represents the task of screen creation. In this way the leaves of the hierarchy are elementary acts which could be interpreted as single operations.

Example: Clips rule performing a simple act (updating of list with settings, used for selection via drop-down list)

a) The starting definition:

```
(deftemplate PK0206
        (slot type       (type STRING)       (default "CB"))
        (slot str_value (type STRING)       (default "0"))
        (slot value      (type INTEGER)     (default 0))
        (slot list       (type STRING)       (default "list_wohnhaft"))
)
```

b) Real time change of the content:

```
(defrule Risikoklassifizierung
    ?fPK0206 <-(PK0206)
=>
    (modify ?fPK0206 (list    "list_Nicht_relevant"))
    (assert (LockFlag2   (value 0.0) ))
```

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 semantic 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

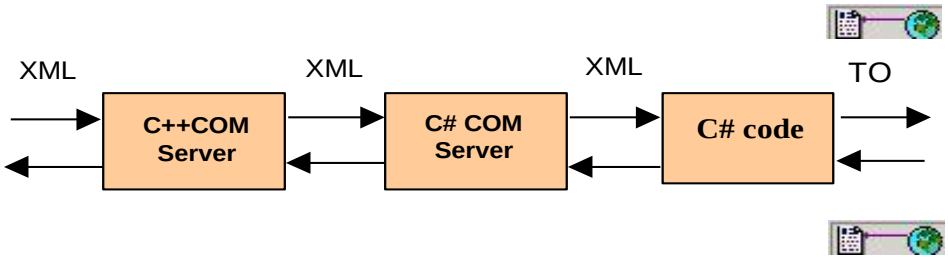

Figure 3 XML based data Exchange

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

With the intelligent client the user can perform the rating procedure in 2 steps:

- Creating models in accordance to the transfer objects (TOs) and saving them with actual data into the database (s. Figure 4).
- Loading the management model to perform the desired steps of the rating process and to see the results (s. Figure 5). The user have only to mark the steps he want to be performed, the communication to the web services starts by pressing the button "Calculate". The results including rating protocol, error messages and calculated ratings and probabilities of default are displayed on the GUI and could be saved into the local database.

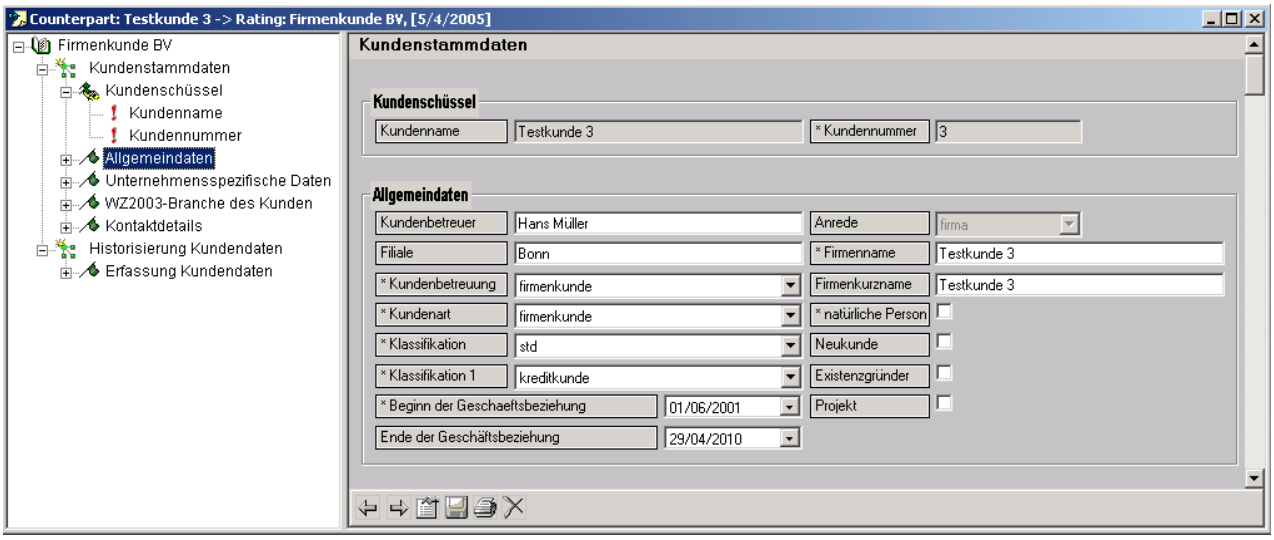

Figure 4 CLIPS model to WSKundeTO

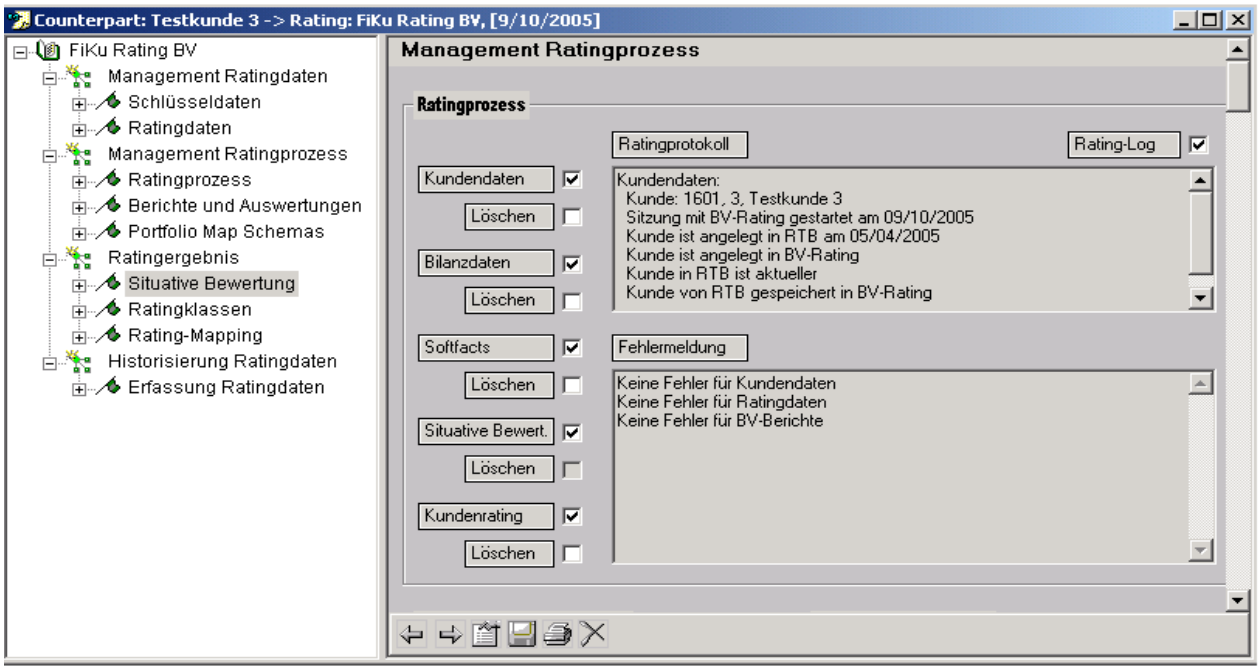

Figure 5 Rating process managing model

CONCLUSIONS AND FUTURE WORK

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

- The rule based GUI is flexible and depends from the model and from the user behavior.
- Counterpart data are exchanged directly between the legacy 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++ and C# Servers can be integrated to another C++ legacy System.

Future development concerns experiments and practical implementations in following directions:

- Deployment of different CLIPS models to map the needs of the users.
- Enhancing the properties of the Implementation and of the 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

[10] Gonzales, A.J., Dankel D.D, “The engineering of Knowledge-Based systems”, Prentice Hall 1993

[11] Thomas C.G, “Design, Implementation and Evaluation of an adaptive user interface”, Knowledge-based systems 6:230-238 1993

[12] Sacerdoti, Earl, “A Structure for Plans and Behavior”, American Elsevier Publishing Company, Inc., 1977

[13] Andre, E., Rist T., “Towards a plan-based synthesis of illustrated documents”, Proceedings Ninth European Conference on Artificial Intelligence, Stockholm, Sweden, 1990

[14] Wahstler W., Andre E, “Plan-based integration of natural language and graphics generation”, Readings in Intelligent User Interfaces, Morgan Kaufmann Publishers Inc., 1998