A Multi-purpose Web Service for Risk Management based on Fuzzy Multi-criteria Ranking

D. Karimanzira, TU-Ilmenau

Abstract— **A multi-purpose web service for risk management is presented in this paper. The approach suggested relies on user supplied data to set the model parameters. However, in real life, people are notoriously inaccurate and unreliable in reporting their preferences. This is especially true as the dimensionality and complexity of the problem increases. Therefore, genetic algorithm techniques are described which can be used to automate the derivation of fuzzy multi-criteria function parameters to consistently find good solutions to reconcile what people say they want with how they act. Several tasks of risk management such as contract allocation, investment portfolios, employee hiring can be solved with the system. The paper focuses mainly on the software technologies applied. Some of the cutting web service (AJAX) and java (EJB3, Hibernate) technologies will be applied.**

***Index Terms*—Risk management, Web service, Hibernate, EJB3, Fuzzy multi-criteria ranking.**

I. Problem

Our client was intending to provide a unique risk management service on line. The business requirement was to provide a compelling user interface to allow people to enter their solution details for advertised projects, employment opportunities, investment portfolios etc. The different solutions/ applicants/ ideas etc., will then be assessed for their risks and decisions made for the best when the application time is due. Lastly, the results and the ranking will be shown for the top list. The task required not only software development expertise but also a lot of the risk management capabilities.

II. Introduction

This work describes software application to risk management. Effective risk management is a complex task. It involves assessing appropriate risk factors and deciding how much risk will be assumed and what actions are warranted to reduce risk [1]. Usually, many of these factors are complex, highly non-linear, and subjective in nature. Thus, by its very nature, risk assessment is highly subjective. There are several solutions to risk management. Fuzzy techniques are especially suited, providing the 'semantic flexibility, representational depth, and evidential reasoning needed to model this general class of problems.

In this paper, we outline a methodology which we have successfully employed to identify and rank options or actions representing a high degree of risk or exposure. This is a classic fuzzy multi-criteria decision problem [3]. Our approach is viable even when the rules underlying the financial system behavior are not well known or are poorly defined. The fuzzy multi-criteria decision technique which we describe allows us to collect subjective data on what analysts perceive are relevant risk factors and their relative importance and to rapidly build individual or group models for risk assessment. This capability is important, because in an organizational context both individual preferences and corporate culture shape how risk is defined [2]. With the techniques we describe in this paper, we are able to study the effects of various points of view and to help to build consensus on a corporate risk model. Additionally, our fuzzy logic tools are computationally fast, and can be easily modified to reflect new conditions and information. The system can be used solely for any allocation problem such as contract allocation, investments, university intake selection etc.

III. Risk assessment and management

The problem we wish to address is how to score and rank options or actions according to some subjective feature that they possess. For example, we might wish to identify "high risk" investment portfolios so that proactive efforts can be taken to manage and/or compensate for the investment risk or decide which construction company is suitable for our construction project.

One way to perform ranking is to build a fuzzy expert system. As described in [12], this type of system contains fuzzy rules which are used to evaluate each candidate and to prescribe a specific course of action or to assign a ranking score. If the risk factors and corresponding actions are not

Manuscript received 30th. November, 2007. This work was supported by a certain Banking Institute.

D. Karimanzira is with the Technical University of Ilmenau and the Fraunhofer Application Centre of the IITB, Karlsruhe, Germany (corresponding author to provide phone: 03677461175; e-mail: divas.karimanzira@ tu-ilmenau.de). Presented at the IMECS 2008 in Hong Kong, 19-21 March,2008

sufficiently well understood to allow definition of a fuzzy rule base, then an alternative approach is to utilize the ordered weighted averaging operator developed by Yager [8]. If the relevant risk factors can be defined as fuzzy sets, then the OWA-Operator can be used to calculate risk assessment scores. This approach is simpler to implement than a fuzzy expert system and can be very helpful in soliciting the user feedback needed to develop a fuzzy rule base. For example, when the effects of competing risk factors are not well understood, the OWA-Operator can be used to easily formulate a variety of risk models which can then be tested and validated. This, in turn, is helpful in elucidating the true underlying risk factors and their interactions.

In the next section we describe in more detail the process of building a risk management system using an OWA-Operator.

IV. Approach

The problem is a multi expert – multi criteria problem. Yager suggests a powerful fuzzy multi-criteria ranking procedure that has been successfully applied in a host of applications [8]. This procedure involves formulating a fuzzy multiple criteria decision model using an ordered weighted averaging operator.

A mapping F from $I^n \rightarrow I$ in [0, 1] is called an OWA operator of dimension n if associated with F is a weighting vector W = $(w_1, w_2, ..., w_n)$ such that W_i in[0,1] and sum(W_i)=1, furthermore $F(a_1, a_2,...a_n) = W_1b_1 + W_2b_2 +... + W_nb_n$ where b_i is the i^{th} largest element in the collection $a_1, a_2...,a_n$

The OWA operator aggregates multiple subjective criteria and assigns an overall evaluation score to a particular alternative under consideration. The first step in the aggregation process is to compute the individual criteria weighting a_j, as shown in (1) below:

$$a_j = \left(\alpha_j \vee p\right) * \left(\left(A_j(x)\right)\right)^{\left(\alpha_j \vee q\right)} \quad \textbf{(1)}$$

where:

a_j = *finalscore*
α_j= *criteria weight*
A_j (x) = *individual fuzzy criteria score n*
$p+q=1$
p = degree of "andness" = 1 - q
q = degree of "orness" =

$$\left(I/(n-I)\right) * \left[\sum_{i=1}^{n}\left((n-i) * w_i\right)\right] \quad (2)$$

n = number of decision criteria
W_i = *weighting vector, as defined above*

The a_j are used in turn, to compute an overall evaluation score. The overall evaluation score is computed according to the definition of the OWA operator given above. The OWA operator is unique in its ability to represent the relative degree of importance of *each* criteria in the decision process, and the overall importance attached to progressively satisfying "more and more" criteria. Note that in this context, "more and more" is a linguistic qualifier representing the decision maker's desire to satisfy all, or most, or half, or some other subjective number of criteria.

The approach suggested by Yager [8] relies on user supplied data to set the model parameters (which include A_j (x), w_i, and a_j, as defined above). However, in real life, people are notoriously inaccurate and unreliable in reporting their preferences. This is especially true as the dimensionality and complexity of the problem increases.

A genetic algorithm technique is used to automate the derivation of fuzzy multi-criteria function parameters to consistently find good solutions to reconcile what people say they want with how they act.

V. SOFTware technology

The technology stack adopted was intended to be cutting edge, whilst not compromising delivery by introducing too much risk. The vision was to have:

- A fully Ajax based user interface, that moved away completely from the old http request/response cycle and only used the asynchronous XMLHttpRequest
- To provide scalability using an EJB3 based form of application servers
- To provide scalability for different tasks of risk management (task dependent user mask)

Fortunately, a number of key technologies had come close to maturity in all these areas and we adopted the following technology stack

- Application Server: JBoss 4
- Persistence Mechanism: Oracle
- CMS System: Hippo [8]
- Object Relational mapping: Hibernate 3 [7]
- Dependency Injection: Spring
- User Interface: Echo 2 [6]

A. Our work process

Whilst this paper is intended to cover the use of these technologies in delivering a fully fledged application, it is worth a brief mention of the work process we had. Our work process is based around rational unified process [4] and we have adopted some of the best tools from the agile movement to help facilitate this (unit testing, uniform build management, continuous integration etc). Whilst many software houses talk about using rational unified process, they tend to fall into two categories:

- Those that pay lip service to the process – but don't actually implement it

- Those that try to follow the process to the nth degree and as a result fail to make progress with kind of speed that customers require.

By contrast we use a strong process that is based around delivering operational slices of functionality to demonstrate progress and to test the requirements. As a results,

- Customer requirements were fully documented using use cases and UML model
- Design was undertaken based on the analysis
- Development was only undertaken on the basis of the design

As a result we were able to deliver an innovative product with fully traceable documentation and design artefacts that mean we can easily support it going forward.

B. User interface

The most extensive and well rounded Ajax type web framework in existence is, for us, echo 2 [5].

We implemented a great prototype of the user interface and were confident of echo 2's ability to deliver. One of the key things about the echo 2 framework is the concept of an application.

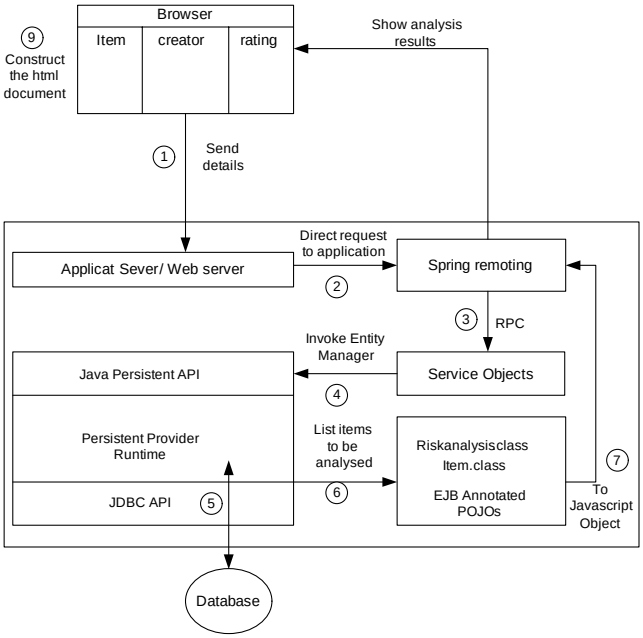

Figure 1: Ajax application architecture with EJB3.0 in JBoss platform

There is one instance of an application per session and what the user sees in the browser is generated by constructing (or changing the state of) a widget lattice that is stored in the application instance.

Whilst using Echo 2 we discovered that whilst it was the most advanced tool for the job we did find that we had to undertake the following:

- Adjust some of the java script in widget peers where it was not quite performing as we expected
- Subclass the echo 2 servlet to ensure that:
  - We can trap non java script type clients and present a "non java script" type version of the page
  - We can present a more polished start up page rather than the default by echo 2
  - Some post back functionality does not work well with IE either under load or restricted bandwidth. Due to the way that IE polls for the post back other events on the browser were being missed.
  - Develop our own widgets where necessary if there was no suitable one available from echo

C. Content management system

One of the customer's requirements was to be able to update the site content using a content management system. We selected the Dutch based Hippo CMS system and built integration between echo 2 and Hippo CMS [7]. Briefly we built a component that could display XHTML fragments and the data to be displayed is pulled from Hippo using DASL.

D. Application service based on EJB3.0 and JBoss4

For the application service, EJB3 and Java 5 and JBoss 4 gave us the optimal solution. All the risk management capabilities were implemented in this layer. We broke down the layers in the EJB space in a manner that should be familiar to most people working in the J2EE space:

- A services facade layer: Consisting of EJB3 Stateless Session Beans providing coarse grain services used by the echo 2 presentation layer and transactional demarcation. All calls from the presentation tier go through this layer. The only EJB's involved in the application are in the service facade layer. All the other layers are implemented using POJO's wired together using Spring.
- An Application services layer containing business components which encapsulate the fine grained business rules needed by the environment.
- Domain Object Managers encapsulate complex rules and relationships between a set of related domain objects.
- A Domain Layer containing Domain Objects mapped to the database using Hibernate.

E. Entity management

We used EJB3.0 annotations backed up by the Hibernate persistence mechanism [6]. This bought the stability of the

mature hibernate product along with some of the extra functionality that Hibernate provides.

Some people could take the view that a multi layer approach as outlined above is no longer necessary in an enterprise scale application. Some question such as the following may arise:

- Why have session beans at all when you could use spring remoting?
- Why have Domain Object managers when you can just make Hibernate calls on a lattice of Domain Objects directly?

Using session beans and spring remoting is arguably more of a style issue than anything else. We went down the Session Bean approach for the following reasons:

- Use of session beans allows for easy integration between JBoss-provided transaction management and the transactional boundaries we wanted for the system
- Use of session beans makes it easy to tune particular services by adjusting the maximum number of beans in the pool for that service.
- We were more familiar with Session Bean remoting than spring remoting

For the domain object managers and hibernate, it is in some ways arguable that, with a sophisticated object relational mapping tool such as Hibernate one can dispense with Domain Object Managers completely. After all, the job of a Domain Object Manager used to be to manage invariants between domain objects in a related logical area, a function now delivered admirably by Hibernate. The argument for the continuing usage of Domain Object Managers is about decoupling. Ideally it would rather be better not have the Application Service Managers know anything about the persistence mechanism. We like to ensure that any calls to the Hibernate framework take place in one area of the code, the Domain Object Managers.

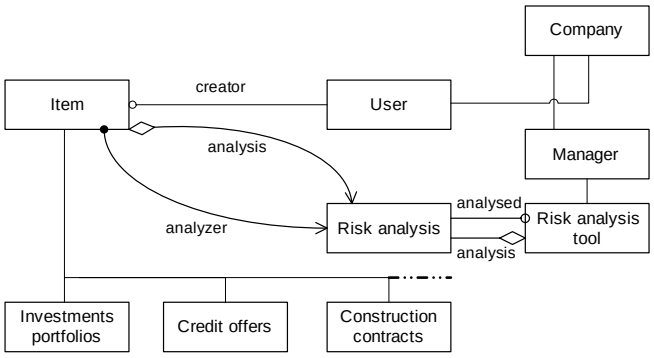

Figure 2: EJB3.0 annotated Persistent Domain Model of POJOs

VI. Business Logic

The business logic is divided into 5 categories, which corresponds to the information processing phases as shown in Figure 3:

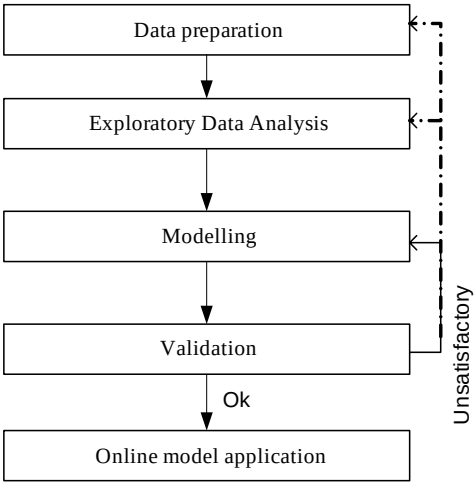

Figure 3: Business logic work flow

A. Stage1: Data preparation and filtering

The first phase involves a careful examination of the data used to measure risk. This is needed to be sure that the data does not contain obvious errors (for example, incorrect date formats, illogical values, etc.). After the data integrity is confirmed, data transformation may also be needed. The transformations may be needed to convert data from continuous to discrete variable (or vice versa) and to scale the data to facilitate subsequent analytical procedures.

B. Stage 2: Exploratory Data Analysis

The second phase involves preliminary data analysis to identify relevant decision variables and to eliminate irrelevant decision variables, pertaining to the measurement of risk. Logistic regression is used to reduce the number of decision variables to a manageable size.

C. Stage 3 and 4: Modelling and Validation

This phase typically involves building a variety of fizzy ranking models using the OWA operator and testing them against a known hold-out sample constructed specifically for the purpose of model validation. A model is refined until it performs to some predefined level of performance or is shown to be unacceptable.

As indicated previously, there are several components to a fuzzy ranking function built using an OWA operator. These include:

- Fuzzy membership functions - these represent a linguistic definition of each decision variable. One fuzzy membership function must be developed for each decision criterion.
- Decision priority weights - these represent the

importance of each decision variable relative to each other.

- Aggregation operators - these operators aggregate the results of the fuzzy ranking function and provide an overall score for a particular individual.

The fuzzy membership functions were derived by interviewing the resident expert and asking him to assign fuzzy scores to selected values. The fuzzy score represents the degree to which the expert agrees or not that the given value is consistent with the linguistic variable to be defined. These (value, fuzzy score) pairs can then be analyzed using automated curve fitting techniques to derive an estimation of the fuzzy membership function equation. Saaty's method of pair-wise comparison is one way to derive the values for the decision priority weights [9]. Saaty's method assumes that the person developing the model can specify the relative importance of each variable relative to every other variable.

In cases where the relative importance of each decision variable is unknown or uncertain, then automated techniques are needed to derive and/or tune the weights. Genetic algorithms were successfully used to automate derivation of the decision priority weights and optimal settings for the aggregation operator. After the components of the fuzzy ranking function are derived, it is a simple matter to substitute them into the ranking equation, to derive an overall ranking score for each option under consideration.

D. Stage 5: Online model application

After the model has been developed and validated, it is used to score, categorize or identify unknown individuals and to make predications about their behaviour from the detailed information they enter online.

VII. Online data processing

After authentification, the users have to select the type of task (investment options, construction contracts, etc) they want to take part. A special mask for each task is displayed to enter the user's own details about the task such as type of material the constructor uses, expenses, maintenance cost for the final building, etc or cost benefits coefficients of an investment.

For every task there is a time limit, which is shown in the browsers. This time limit is the trigger for the task dependent calculation for risk and ranking as outlined in the previous section. For each project the results are summaries and the identification numbers of the top 5 are listed on the web page.

The EJB component model adds a number of important component-model features that make it even more appealing as a middleware component, such as transactions, security, and database connectivity.

Transaction points and data access points often surface around business logic components in a distributed system. These aspects make EJB a good technology to use for middleware business-logic components with several execution phases and access points. For example, depending upon our business needs, we can call EJB with the business logic remotely from the presentation tier (e.g. Data collection and output logic) or locally from other plain-old Java objects (POJO) on the business tier (e.g. Modelling and validation logic), as Figure 4 illustrates.

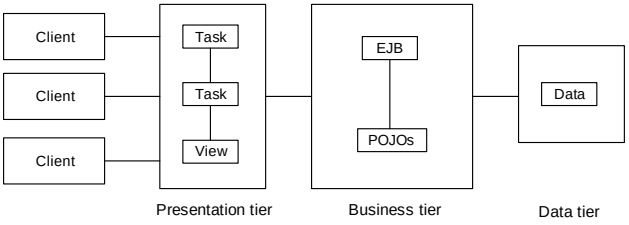

Figure 4: Calls to EJB

The flexibility of EJB access enables you to employ the composition principles to build logical components out of multiple JavaBeans and one or more POJO.

VIII. Conclusions

In this paper we described a method to perform risk assessment using a fuzzy OWA operator. Additionally, we discussed how genetic algorithms can be used to automatically determine the parameters of a fuzzy multi-criteria ranking problem. This is a practical approach for real life problems, offering an alternative to fuzzy rule based systems for risk assessment.

Our client has been able to be delivered a fully featured Web 2.0 style enterprise application. The technology stack of echo 2, JBoss, EJB 3 and provides a powerful tool for providing scalable and reliable web based application with rich user interface.

Acknowledgment

We would like to thank all partners and sponsors of the project, especially the economists involved in designing the rule-bases for different risk management tasks.

References

[1] Bernstein, Peter, *Portable **MBA** in Investment*, John Wiley & Sons, copyright 1995, pp. 243-307.
[2] Earl Cox, *A Fuzzy Systems Handbook, AP* Professional, copyright 1994, pp. 435-469.
[3] Rubinson, Teresa C. ***A Fuzzy Multiple Attribute Design and Decision Procedure for Long Term Network Planning,*** Ph.D. Dissertation, Polytechnic University, June, 1992.
[4] Arlow, Neustadt: UML 2 and the Unified Process, Addison-Wesley Professional, 2005
[5] http://www.nextapp.com/platform/echo2/echo/doc/tutorial 18.11.2007
[6] http://www.hibernate.org/hib_docs/reference/en/html/index.html 18.11.2007
[7] http://www.hippocms.org/display/CMS/Reference 18.11.2007

[8] R. R. Yager, "On ordered weighted averaging aggregation operators in multi-criteria decision making," *IEEE Trans. Syst., Man, Cybern.*, vol. 18, pp. 183-190, 1988.
[9] Saaty, T., "Hierarchies, priorities, and eigenvalues," University of Pennsylvania, Philadelphia, 1972.
[10] Falkenauer, Emanuel (1997). *Genetic Algorithms and Grouping Problems*. Chichester, England: John Wiley & Sons Ltd. ISBN 978-0-471-97150-4.
[11] Goldberg, David E (2002), *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*, Addison-Wesley, Reading, MA.
[12] Otto, P., Wernstedt, J. (2002). Fuzzy methods for the optimal emulation of expert decision behaviour and the optimal controller design. In: Proceedings of the East West Fuzzy Colloquium 2002, 10th Zittau Fuzzy Colloquium, pp. 351–367.