2009

A Fuzzy Logic Based Approach For Enterprise Application Evaluations

Jan Stefan Addicks
OFFIS – Institute for Information Technology, addicks@offis.de

ENTERPRISE SYSTEMS

A FUZZY LOGIC BASED APPROACH FOR ENTERPRISE APPLICATION EVALUATIONS

Addicks, Jan Stefan, OFFIS – Institute for Information Technology, Escherweg 2, 26121 Oldenburg, Germany, addicks@offis.de

Abstract

This contribution presents an approach to use fuzzy logic and fuzzy rule systems for enterprise application evaluations (enterprise architecture dependent application evaluations). Such evaluations regard all relations to enterprise architecture artifacts in order to evaluate an application. There may be a large number of criteria to be considered and some of them have to be assessed by humans, which may introduce vagueness to the results. We present a method to assign the results of criteria to linguistic terms that have semantics like “good” or “poor”. Since it is often relevant to have exactly one characteristic to compare applications to each other, a fuzzy rule system approach is presented that aggregates those linguistic terms for each criterion of an application to an overall indicator of an application’s quality.

Keywords: *Enterprise Architecture, Evaluations, Metrics, Fuzzy Logic*

1 INTRODUCTION

Nowadays organizations run numerous applications that support their everyday business. The number of applications that directly support the business continuously grows, as does the number of relations between applications. Thus, stakeholders (like IT-architects or chief information officers (CIO)) are faced with a growing complexity of application landscapes (denoted as landscape in the following), which are defined as the entirety of all applications and all relationships between them (cf. Hess, Humm, and Voß (2006), Wittenburg (2007)).

In order to fulfill business requirements, organizations must improve the quality of landscapes. Therefore, a well documented application landscape is essential for corresponding stakeholders to be able to manage the current and plan the future state. There are approaches that focus on modeling and documenting the landscape and their interconnections with the hardware infrastructure as well as with the business elements (cf. Frank 2002, Lankhorst 2005, or Winter and Fischer 2006). Besides these results from academia, commercial tools to support these issues exist. An overview of major tools can be found in (Technische Universität München, Chair for Informatics 19 (sebis) 2008, James 2008, or Peyret 2007). Since landscapes are part of enterprise architectures (EA), these tools generally also address or focus on EA as a whole.

Nevertheless, in some disciplines, like IT consolidation, a well documented landscape is still insufficient to support the CIO. When two or more applications with similar functionality have to be compared to each other in order to determine the one that best suits the landscape, other instruments must be used. Metrics as well as assessment and evaluation methods seem to be suitable here. So far, no established methods exist, but several approaches address the issue of assessments and evaluations of enterprise applications and application landscapes (cf. Addicks 2009, Durst 2006, Gammelgaard 2007, Gammelgaard, Simonsson, and Lindström 2007, Lankes and Schweda 2008, or Lankes 2008).

These approaches differ from classical software metrics (cf. Fenton & Pfleeger 1998 or Kazman, Klein, & Clements 2000), since the latter are used to evaluate the quality of single software systems within the

software engineering phase and explicitly do not regard the environment in an organizational context. That aspect is important for evaluations since properties of applications can influence further applications' properties. For instance, the response time of an application can depend on the response time of another. Besides application dependencies, the quality categorization of applications could depend on other aspects, like strategically decisions or regulations from authorities.

Enterprise application metrics (application landscape metrics) are not used commonly in practice so far, but according to a survey (cf. Lankes 2008), the surveyed organizations are considering the use in the future. Thus, there are only few application landscape metrics in literature yet; for instance, Lankes (2008) has defined and evaluated a concrete metric to assess the fault propagation within a landscape. Due to the importance of the topic, we consider that further enterprise application metrics will to be published soon.

In order to compare two or more applications, it is often not sufficient to apply a single metric, since in many cases several different criteria play a role. Due to the high quantity of applications and relations between them as well as the multitude of criteria to be regarded, a methodology as well as a tool to support enterprise application evaluations is required. In our previous work we present such a methodology (Addicks 2009, Addicks and Steffens 2008); however, how key figures resulting from several metrics for a concrete application are combined to a single indicator of the application's quality remains an open question.

We address this issue and present an approach to support enterprise application evaluations here. The approach bases on fuzzy control language (FCL) to manage vague expectations of CIO and other related stakeholders. We consider the underlying fuzzy logic as suitable for evaluations if vague relationships between measured values and expected evaluation terms exist. Stakeholders applying this approach are able to use linguistic terms that are more intuitive than numerical values.

The remainder of this paper is structured as follows: The following section introduces some relevant terms for enterprise application evaluations. A presented diagram will additionally visualize the relations between those terms. In section 3, the context of enterprise application evaluations is introduced. While the fourth section gives basic information on fuzzy logic, the fifth section presents the fuzzy control language, which bases on fuzzy logic. FCL is used to infer overall indicators for an application's quality regarding its landscape context. After that, the sixth section briefly presents the underlying evaluation method, before the following section states a brief example scenario of the approach's practical utilization. The last section gives a brief summary and presents future work.

2 TAXONOMY

Before we present our approach we will introduce some basic terms in order to give a better understanding of this contribution and the overall context. Figure 1 visualizes the terms mentioned below and their relationships to each other. This information model is based on (Addicks and Steffens 2008); however, it is only an extract of the overall information model for our enterprise architecture dependent application evaluation approach (cf. Addicks 2009). We consider this extract to be adequate for the scope of this paper.

The basic term for the presented approach is *artifact*. An artifact is a business relevant object of the enterprise. Artifacts are connected to each other via *relations*. Artifacts as well as relations have *attributes* like an identifier or a title. An important term is *application*, which is a software technical artifact. The composition of all artifacts and their relations is the *enterprise architecture*. With this definition we differ from other definitions like the one of Lankhorst (2005), since we distinguish *enterprise architectures as discipline* from *enterprise architecture as architecture*.

In order to evaluate applications the CIO has to specify *assessments* which contain all *criteria* to be regarded. Criteria can be versatile and may focus on all artifacts of the enterprise architecture. For example criteria that evaluate applications regarding the number of business processes they support

may be chosen. Furthermore some criteria evaluate applications regarding monetary aspects like their maintenance costs or license fees but also regarding levels of compliance with rules and regulations from authorities (cf. Addicks 2009).

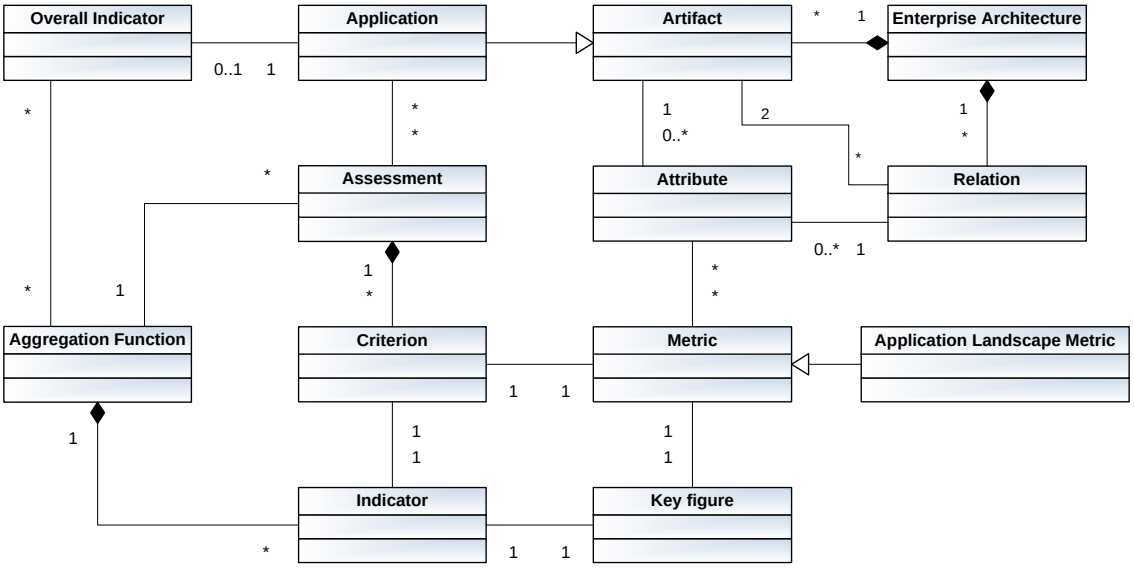

Figure 86. Information model of relevant terms

Criteria are assigned to *metrics*, which are used to calculate *key figures* for each criterion regarding a certain application. For example, a key figure might be "80%" for an application a_1 regarding the concrete criterion "availability". Key figures are assigned to *indicators,* which are linguistic representations of key figures. The key figure "80%" for "availability", as an example, can be assigned to the indicator "average". Indicators belong to criteria, which have specifications of mappings from key figures to indicators. Specializations of metrics are *application landscape metrics* that regard all kinds of artifacts of the enterprise architecture to calculate a key figure of an application.

Since it is often important to have exactly one characteristic to compare two or more applications to each other, all indicators of a certain application have to be aggregated. This aggregation is performed by an *aggregation function*. An aggregation function's result is an *overall indicator* for the application's quality. A concrete aggregation function is presented in section 6. The presented aggregation function delivers a linguistic overall indicator. With a semantic like "good", such an output value becomes interpretable by our method as well as by people.

3 CONTEXT OF ENTERPRISE APPLICATION EVALUATIONS

Due to the increasing number of heterogeneous applications, interwoven by numerous relationships, many organizations are faced with a challenging complexity. IT architects as well as CIO have to manage the landscapes. They have to plan the to-be architectures and have to decide whether applications should be substituted, removed, or added to the landscape.

In order to support the management of the enterprise architectures, a growing number of organizations buy or implement Enterprise Architecture Management (EAM) tools. These EAM tools (cf. James 2008, Peyret 2007, Technische Universität München, Chair for Informatics 19 (sebis) 2008) cover analysis and reporting functions, data import, and data export of the whole or parts of the enterprise architecture (EA) in general. One part of an EA is the documentation of the landscape (Lankhorst 2005). Common functionality regarding landscapes usually covers the analysis and visualization of interdependencies between individual applications as well as their connections with logical business elements and the underlying technical infrastructure. Evaluations of applications or landscapes, however, are hardly

supported in an appropriate manner. As a result, whenever a new IT project is initialized, one has to analyze and document the current state manually.

In preparation of the procurement of an application, or for the conceptual design of a piece of individual software respectively, CIOs have to compare all possible alternatives and weigh the requirements to find the best fitting application for the existing landscape, or at least check for properties which do not fit with critical requirements. For example, if the recoverability of the whole landscape is critical for the business, an application that does not allow for fault-recovery can cause high expenses. CIO ought to regard all relevant requirements and decide whether the new software is appropriate for the existing landscape or not. Due to the complexity resulting from the high number of applications and interconnections, this task is nearly impossible without adequate support. Today, in larger software integration projects, time-consuming as-is analyses are generally performed up-front. Such analyses take much effort and the results will often not be used again after the end of the project.

Furthermore, enterprise application evaluations are required within IT consolidation projects. After mergers or acquisitions the application landscape contains several redundant applications. Redundant applications are these applications that are different but have similar functionality. IT consolidation projects have the objective of removing such redundancy in order to reduce the overall complexity, to reduce interfaces between applications, and to increase the consistency of data. The consistency of data is getting down whenever it is impossible to fully synchronize data modifications between redundant applications and thus at least one of them has out-of-date data. Enterprise application evaluations (EA dependent application evaluations) support CIO to compare redundant applications and therefore identify these that might be removed.

We consider integrating our approach into an EAM tool since we anticipate synergy effects from such a liaison. EAM tools can use indicators as operating figures for analyses and reports, and evaluations implementing our approach benefit from documented landscapes, application data, and the possibilities to visualize results (cf. Wittenburg 2007).

4 FUZZY SETS

Fuzzy set theory was introduced by Zadeh (1965) in 1965. In classical set theory, every element x is either completely in a set s ($x \in s$) or not ($x \notin s$); however, fuzzy sets do not have such sharp borders. In fuzzy logic, every element x is allowed to be in a set s partially. A responsible membership function $\mu_s(x)$ does not only assign 0 ($x \notin s$) and 1 ($x \in s$) to an element *x*, but additional all real numbers in-between. Thus, the result of a membership function $\mu_s(x)$ expresses in which degree x belongs to *s*. Zadeh (1965) adapted classical set operations for fuzzy sets, like complement, intersection, and union.

Fuzzy set theory is similar to the way humans think, thus it is especially suitable for matching measured numerical values to vague expectations of individuals interpreting these values. Let a set *cheap* have the following membership function (1):

$$\mu_{cheap}(x) = 1; \quad x \in IR^{<100} \tag{1}$$

This function assigns all values which are less than 100 to the set *cheap*. What about a price of 101? One would say that this value is “almost cheap”. To express that in classical set theory we need to define a second set “almost cheap”. Fuzzy logic offers the possibility of defining a membership function that assigns some values that are outside the set to the set with a certain degree. Figure 2 visualizes the membership functions for this example. The left one represents the membership function for classical set theory, whereas the right one represents the fuzzy one. The filled area under the right graph indicates values that are assigned to the set *cheap* with a certain degree.

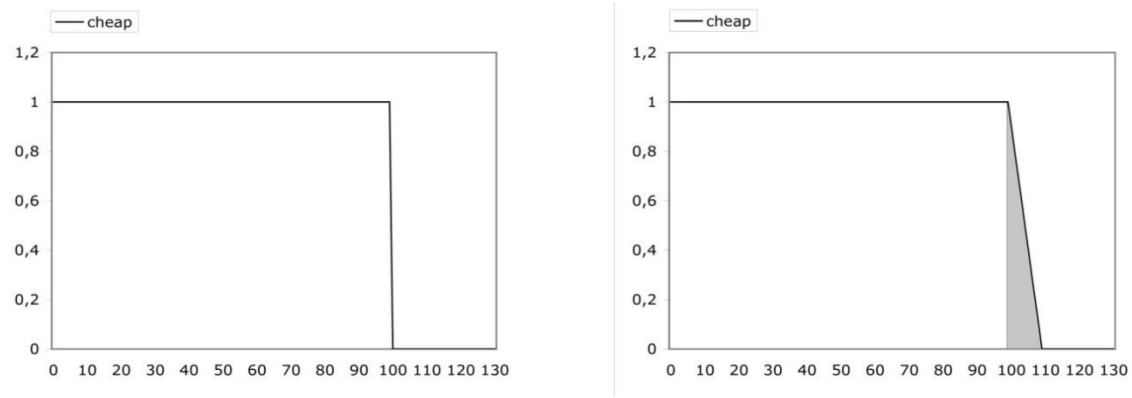

Figure 2. Graphs for membership functions for classical (left) and fuzzy (right) set theory

A similar way of assigning values to evaluation sets is applied within the presented approach. Measured or defined values which would be marginal outside a defined set in principal are assigned to that class due to well defined membership functions. Hence, we consider the results to be more adequate.

In the following paragraph we present some relevant terms which are popular in fuzzy logic. Sets in fuzzy logic are frequently used to describe a term in natural language. Examples are terms like *good*, *old*, or *cheap*. In fuzzy logic, those sets are called *linguistic terms*. We consider such linguistic terms as results of evaluations more suitable for stakeholders than numerical values. Hence, our approach will use measured numerical values or defined percentages for applications' properties as basis and infer linguistic terms like *good*, *bad*, and *average* by using fuzzy technology. The conversion of numerical values into a degree of membership to certain linguistic terms is called *fuzzification*. Complementary, the conversion of the degree of membership to linguistic terms into a numerical value is called *defuzzification*. Further definitions are given in (IEC 61131-7).

5 FUZZY CONTROL LANGUAGE

In this section we introduce fuzzy control language (FCL) which was standardized by the International Electrotechnical Commission (IEC) in 1997 (IEC 61131-7:2000 - Programmable controllers - Part 7: Fuzzy control programming) (IEC 61131-7, 1997). FCL can be used to describe fuzzy sets (Zadeh 1965) as well as fuzzification, defuzzification, and fuzzy rules. Practical application of fuzzy logic and the fuzzy control language can be found in programmable controllers for certain hardware. Saritas et al. (Saritas, Etik, Allahverdi, & Sert 2007), for instance, present a control system for operating room air-conditioning based on a fuzzy expert system. A typical FCL definition contains three parts:

- Declaration of input and output variables
- Definition of fuzzifications and defuzzification
- Definition of rules that control the defuzzification

In the following paragraph, we present an extract of an FCL description example. Please note that the given line numbers are only for explanation and not part of the FCL syntax. First of all, the involved variables have to be defined. The example in listing 1 shows the definition of three input variables (*availability*, *performance*, and *bio_criticality*) and an output variable *score*. The latter is used as indicator regarding the application's quality. All variables are defined as real numbers.

```
1: VAR_INPUT
2:          availability          : REAL;
3:          performance           : REAL;
4:          bio_criticality       : REAL;
5: END_VAR
6: VAR_OUTPUT
7:   score : REAL;
8: END_VAR
```

Listing 1. Description of input and output variables in FCL

After specifying the input and output variables the next step is to define the linguistic terms for each variable and the related membership functions. Listing 2 presents the corresponding definition for the variable *availability*. The other variable definitions are similar. In listing 2, line 2, the linguistic term *poor* is defined. The three tuples on the right-hand side define the related fuzzy membership function. Each tuple represents a point of the functional graph within a coordinate system. If the linguistic variable is less than the first point, the membership degree is defined as the first point's degree. Accordingly, if the linguistic variable is greater than the last point the associated degree is assigned.

```
1: FUZZIFY availability
2:   TERM poor          := ( 0,1) (50,1) ( 75,0);
3:   TERM average       := (70,0) (80,1) ( 95,0);
4:   TERM good          := (90,0) (95,1) (100,1);
5: END_FUZZIFY
```

Listing 2. Definition of membership functions for a variable in FCL

Figure 3 represents the graphs of the membership functions which are defined in listing 2. We consider the use of such a graphical representation as an interface for adapting the membership functions. Changing the functions by dragging and dropping the associated points and thereby redefining the function graphically ought to be more intuitive than textual definitions (cf. listing 2).

The definition of an output variable *score* is shown in listing 3. Linguistic terms and their membership functions for output variables can be defined in a similar manner to that of input variables (cf. lines 2 to 4). The defuzzification can also be specified. In this example we use the center of gravity (COG) method (line 5). This method calculates the numerical value for a linguistic term by using the center of the area of the membership function and uses its value for the defuzzification.

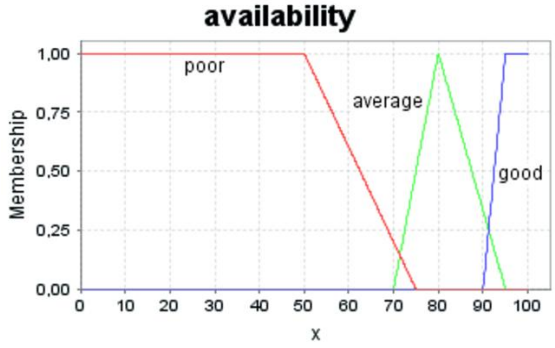

Figure 3. Graphical representation of the membership functions from listing 2

In (IEC 61131-7, 1997, Kecman, 2001) COG and other methods are presented in detail. Line 6 in listing 3 contains the definition of a default value, which is returned if no fuzzy rule (see below) is activated.

```
1:  DEFUZZIFY score
2:    TERM poor              := (  0,0) ( 7.5,1) (  15,0);
3:    TERM average                   := (7.5,0) (  15,1) (22.5,0);
4:    TERM good              := ( 15,0) (22.5,1) (  30,0);
5:    METHOD      : COG;
6:    DEFAULT                := 0;
7:  END_DEFUZZIFY
```

Listing 3. Definition of a defuzzification method in FCL

Listing 4 gives an example of the definition of rules in FCL. The rule definition's structure is comparable to those used in rule based systems and expert systems (cf. e.g. Hayes-Roth 1985). In line 2 the algorithm for the operators AND and OR are specified. To fulfil de Morgan's Law, the algorithms for AND and OR is used pair-wise (cf. IEC 61131-7). In our case we use the *minimum algorithm MIN* (MIN = Min ($\mu_1(e),.., \mu_n(e)$)) for the AND-operator and the *maximum algorithm* for OR. Further algorithms can be found in the specification. In line 3 the activation method is defined which is based on the minimum algorithm (see above) in this case.

Every rule has an identifier, a condition block, and a conclusion block. The condition block contains variables and linguistic terms. The concatenation of those pairs and the operators AND, NOT, and OR as well as parenthesis build up the assertions of a rule, which control the conclusion block. The conclusions assign a linguistic term to the output variable. In line 4 and 5, for instance, a fuzzy rule is defined. It assigns the term *poor* to the output variable score if the variable availability is fuzzified to the term *poor* OR the variable performance is fuzzified to the term *poor* (cf. listing 1). Note that the OR operand uses a selected algorithm and is not a Boolean OR (IEC 61131-7). It is possible to explicitly *not* assign the output to a certain term (see line 7); furthermore, the degree of membership, as the conclusion of a rule, can also be adapted by using a weighting factor (cf. line 11).

We consider the definition of such rules as intuitive. Stakeholders who are not willing to use and understand algorithms or program source code are able to express interrelation between linguistic terms.

```
1:  RULEBLOCK No1
2:          AND : MIN;
3:          ACT : MIN;
4:          RULE 1: IF availability IS poor OR performance IS  poor
5:        THEN score IS poor;
6:          RULE 2: IF availability IS NOT poor AND bio_criticality IS NOT poor
7:        THEN score IS NOT poor;
8:          RULE 3: IF availability IS good AND bio_criticality IS very_good
9:        THEN score IS good;
10:         RULE 4: IF bio_criticality IS average AND performance IS poor
11:             THEN score IS poor WITH 0.5;
12:       END_RULEBLOCK
```

Listing 4. Rules in FCL that control the defuzzification

6 FUZZY LOGIC BASED EVALUATIONS OF ENTERPRISE APPLICATIONS

In this section we briefly present our fuzzy logic based evaluation method, which extends the one described in (Addicks and Steffens 2008). We explicitly focus on the aspects of fuzzy aggregation and do not explain the general method in detail. To infer an overall indicator for the quality of a single application, the CIO has to define relevant criteria, which connect certain application attributes as well as other aspects of the application's context. An example of such a criterion is *BIO criticality*, which is covered in detail in the next section. All criteria must be defined as FCL input variables, as seen in listing 1.

The CIO defines fuzzy membership functions for each criterion (cf. listing 2). These membership functions are used in the fuzzification process to determine the fuzzy sets and the related degree of membership. Since these sets are linguistic terms like *good*, *poor*, or *average*, the results represent quality indicators for criteria. The difficulty of the exact specification of membership functions varies from criterion to criterion as well as from organization to organization. When there are no reference values, like the minimum threshold of an application's availability warranted in some service level agreements, the function has to be defined based on expert knowledge.

All indicators have to be aggregated to determine an overall indicator for an application. The fuzzy rules mentioned above (cf. listing 4) can be applied for this purpose. The condition parts of each rule must contain assertions that consist of input variables and related fuzzy terms like "availability IS good" (cf. listing 2). The overall indicator is set to one of the specified terms in the rule's conclusion block (cf. listing 3). As a foundation, the overall indicator must be defined as an FCL output variable, like *score* in listing 1. The CIO has to specify under which condition the indicator has which value (is assigned to which fuzzy set to which degree of membership) by defining adequate rules. Tool-support is required to support the rule definition process and to preserve the integrity of the rule base.

By applying the fuzzy rules, the defuzzification process determines the output variable. This variable is a linguistic term and therefore a good indicator of the application's quality. The overall indicator can be represented graphically using software maps (Wittenburg 2007). For example, each application of the landscape could be visualized as a rectangle containing the application's name while the overall indicator is represented by a small symbol, for instance a traffic light, placed next to the corresponding rectangle. The traffic light representation fits perfectly, if there are exactly three different terms for the output variable.

In our method every criterion is defined as a module (cf. figure 4) that may be used as a building block for an assessment. This allows for flexibility regarding assessments since every organization can apply criteria deemed relevant and may furthermore exchange criteria with ones that are more appropriate.

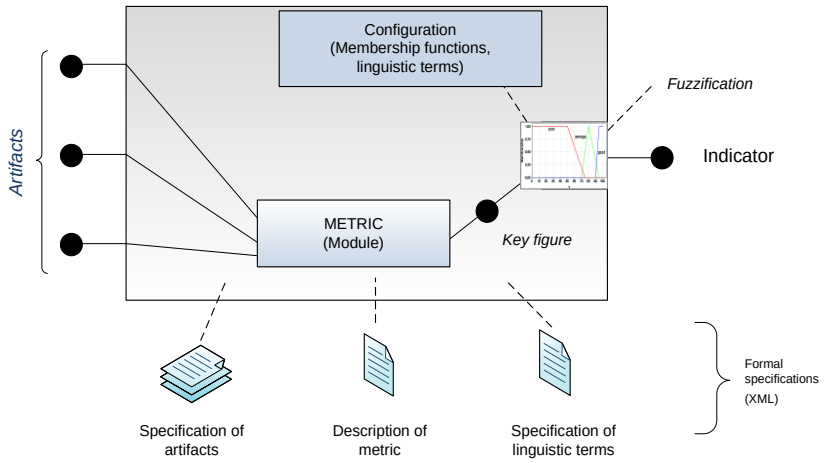

Figure 4. Module representation of a criterion

A criterion has artifacts as inputs and an indicator as output. Each criterion has a concrete metric that uses the input-values to calculate a key figure. A configuration is added to the criteria in order to specify the linguistic terms for the indicator as well as to specify the membership functions (cf. section 5). By using the specified membership functions, the fuzzification part of the module maps the key figure from the metric to the indicator (output value). To avoid a loss of information, the degree of membership is assigned to indicators.

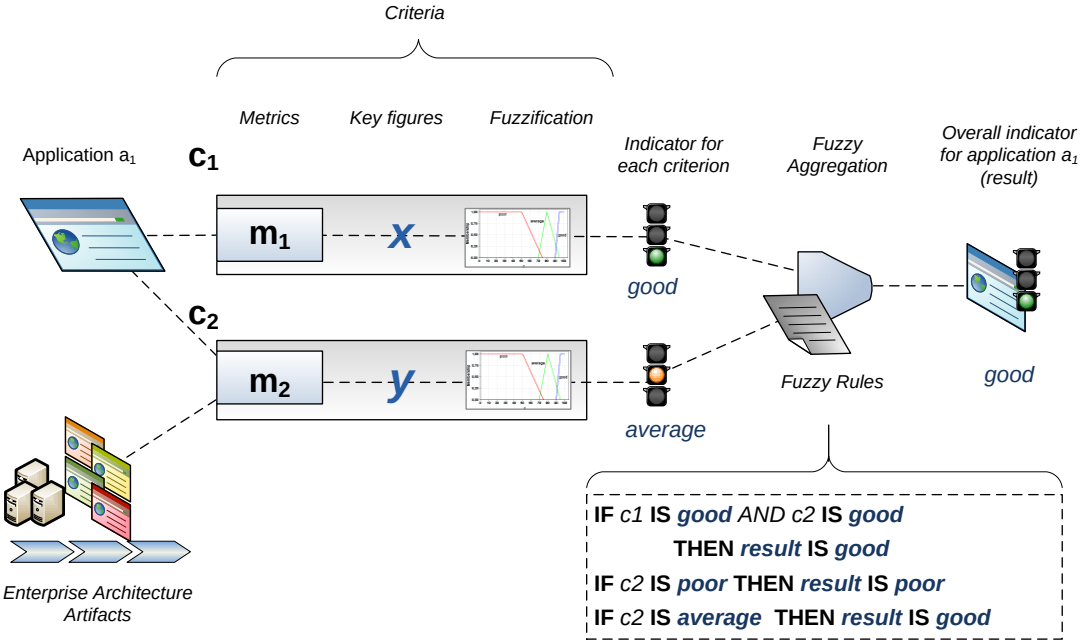

Figure 5. Fuzzy logic based evaluation of enterprise applications

The fuzzy evaluation method is depicted in figure 5. The evaluation method to infer an indicator for a landscape extends the previously presented method. The differences are minor and not relevant for the scope of this contribution and we omit further explanation.

7 EXAMPLE SCENARIO

In a project with an industrial partner, we had the task to generate an IT city plan of the partner's landscape. The primary task was to gather information on a selected set of applications. In the early project phase this chosen view was of particular importance. One type of Information of explicit interest was which applications manage which *business information objects* (BIOs) and which application has the

data sovereignty over a certain BIO. Applications having the data sovereignty over a BIO are those applications (the *leading* application for a BIO) where all modifications of instances of the BIO have to be committed. A BIO instance is a concrete occurrence, for example a customer *city bank* as instance of the BIO *customer*.

Defining the leading applications for BIOs is relevant for landscapes since it offers a possibility to check for potential data inconsistencies. If there are non-leading applications for a certain BIO that are used to enter or modify data for BIO instances, inconsistencies within the whole landscape regarding that information may result. To avoid this, interfaces have to be established in order to realize automatically data transfers of such BIO instances with the leading application.

Our task was to check applications regarding this particular aspect. We defined a criterion *BIO criticality* and a metric to determine appropriate values. We require information on which BIO is managed by which applications, which application has the data sovereignty over which BIO and which BIOs are being transferred via defined interfaces and data transfers. This criterion is a good example of the importance of landscape dependent evaluations are in contrast to using software metrics to determine one applications quality in the enterprise context.

The following figure 5 depicts one of our graphical representations with an extract of falsified data. That visualization type is a table. The applications of the landscape are visualized in the header row, and all BIOs of the landscape in the header column. A cell contains a grey element when the corresponding application uses the corresponding BIO. A '+'-symbol is added if the application is leading for the BIO. (Directed) data transfers are represented as arrows. The arrows are solid when the transfer is automatic and dashed when the transfer is executed manually.

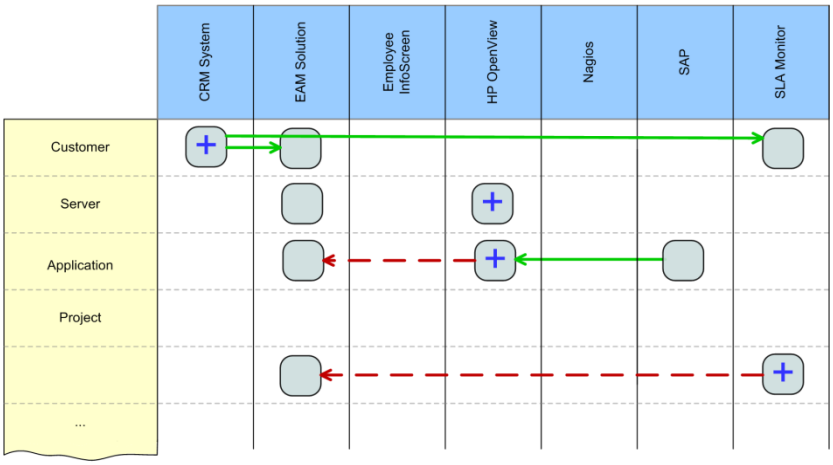

Figure 6. BIO - application overview map (falsified data)

Given that information we can use a metric to determine values for each application regarding the criterion *BIO criticality*. The BIO criticality C_{BIO} of an application *a* depends on four aspects with different weighting factors: $\alpha, \beta, \gamma,$ and δ. Each aspect uses a particular function as listed below:

- $data_s(a)$*:* returns the set of all BIOs over which *a* has the data sovereignty and therefore *a* is the leading system
- $transfer_{auto}(a)$*:* returns a set of all BIOs which are used by *a* and which are synchronized with the BIO's leading system automatically
- $transfer_{man}(a)$: returns a set of all BIOs which are used by *a* and which are synchronized with the BIO's leading system manually
- $data\ (a)$*:* returns a set of all BIOs which are used by *a* and which are not synchronized with the BIO's leading system

The complete metric uses the number of the elements in each set and multiplies it with one of the above mentioned factors. The sum of all values is the result for application *a* regarding that criterion. The metric is represented by the following formula (2).

$$C_{BIO}(a) = \left| data_s\ (a) \right| * \alpha + \left| transfer_{auto}(a) \right| * \beta + \left| transfer_{man}(a) \right| * \gamma + \left| data\ (a) \right| * \delta \qquad (2)$$

The results of this metric can be assigned to fuzzy sets representing quality indicators for the criterion *BIO criticality*. The definition is similar to the one presented in listing 2. In our example, the landscape's applications have to be evaluated regarding three criteria: *performance*, *bio_criticality*, and *availability*. We do not want to define the metrics for availability or performance since those metrics are frequently discussed. An overall indicator for an application can be inferred by applying the fuzzy rules concept presented in section 5 (cf. listing 4). The membership functions for the input values are defined by stakeholders regarding their own experiences or by using existing definitions from the organization. For instance, there are service level agreements that guarantee a maximum down time for an application to customer. That value is adequate to define a membership function for the criterion *availability*. Such or similar thresholds are considered to provide a good information base for defining membership functions.

The prototype of our evaluation tool that is used for inferring overall indicators for applications bases on jFuzzyLogic (cf. *http://jfuzzylogic.sourceforge.net*). jFuzzyLogic is an open source fuzzy logic package written in java that supports FCL. For this scenario we use the previously presented FCL definitions. The values for the input variables *performance* and *availability* can result from monitoring tools that monitor the applications. Figure 6 contains a screenshot from our tool that contains a falsified extract of data. It depicts the used attributes and their indicators as well as the application's overall indicator. The diagram in the lower right of that figure visualizes the membership functions for the defuzzification.

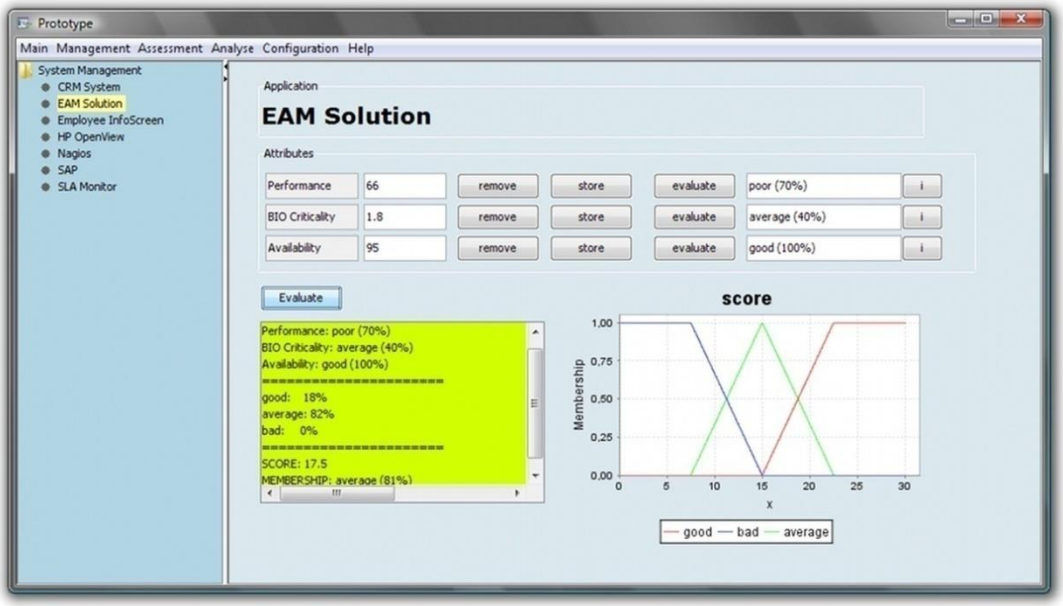

Figure 7. Evaluation of an application using FCL (screenshot containing falsified data)

We use linguistic terms in order to make application assessment transparent for stakeholders. For people it might be more intuitive if we use terms like *good*, *poor*, or *average* instead of values like "12", "50%", or "1.25".

With the declarations and definitions from the above presented listings we have performed an exemplary evaluation with a small set of applications. Each application has been rated according the linguistic terms which represent the natural language semantic. Since visualization are often more

expressive than textual representations, we map the indicators of each application with a certain color. We assign the colors *green*, *yellow*, and *red* to the linguistic terms *good*, *average*, and *bad*. Furthermore, as an additional feature, we map the membership degrees the terms to the RGB-components of those colors. The quality of an application is represented by gradual nuances of the colors from *green* (*good*) over *yellow* (*average*) down to *red* (*bad*). Those colors can be used to colorize the graphical representations of applications in landscape visualizations, like software maps (Wittenburg 2007).

8 CONCLUSION AND FUTURE WORK

This paper presents an approach to the use of the fuzzy control language to infer overall indicators for the quality of applications regarding their landscape. This approach uses fuzzy logic in order to handle the vagueness of certain application properties regarding their assignment to classification sets. We introduced the overall context first and presented fuzzy logic as well as the fuzzy control language next. The practical application of this approach was shown by a brief example. The use of linguistic terms is considered more intuitive for stakeholders.

The presented approach uses FCL and fuzzy logic, which is one possibility to implement the above-mentioned aggregation function. In the future, further alternatives can be implemented and compared to the one given.

We are currently setting up a project with another industrial partner with the primary objective of evaluating whole application landscape with a few hundred applications. To identify the criteria to be considered, workshops with employees from our partner are to be performed. This project will serve as an extensive scenario to evaluate the complete method and the presented fuzzy logic approach.

References

Addicks, J. S. (2009). Enterprise Architecture Dependent Application Evaluations. In Proceedings of the 3rd IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST 2009). Cyber Engineering and Creating Value by Making Connections. Istanbul.

Addicks, J. S., & Steffens, U. (2008). Supporting Landscape Dependent Evaluation of Enterprise Applications. In M. Bichler, T. Hess, H. Krcmar, U. Lechner, F. Matthes, A. Picot, et al. (Eds.), Multikonferenz Wirtschaftsinformatik, MKWI 2008, Proceedings, 1815–1825. GITO-Verlag, Berlin.

Durst, M. (2006). Kennzahlengestütztes Management von IT-Architekturen. In H. P. Fröschle & S. Strahringer (Eds.), HMD: 43.2006, 250. IT-Governance (pp. 37–48). Heidelberg: dpunkt-Verl.

Fenton, N. E., & Pfleeger, S. L. (1998). Software metrics: A rigorous and practical approach, 2nd ed., PWS Publ., Boston

Frank, U. (2002). Multi-perspective enterprise modeling (MEMO) conceptual framework and modeling languages. Proceedings of the 35th Annual Hawaii International Conference on System Sciences, 2002. HICSS, 1258-1267.

Gammelgaard, M. (2007). Business Value Assessment of IT Investments: An Evaluation Method Applied to the Electrical Power Industry. Dissertation, Royal Institute of Technology (KTH), Stockholm.

Gammelgaard, M., Simonsson, M., & Lindström, Å. (2007). An IT management Assessment Framework: Evaluating Enterprise Architecture Scenarios. Information Systems and E-Business Management, 5(4), 415-435.

Hayes-Roth, F. (1985). Rule-based systems. Communications of the ACM, 28(9), 921-932.

Hess, A., Humm, B., & Voß, M. (2006). Regeln für serviceorientierte Architekturen hoher Qualität. Informatik Spektrum, 29(6), 395-411.

IEC 61131-7 (1997). IEC 61131-7: Programmable Controllers

James, G. A. (2008). Magic Quadrant for Enterprise Architecture Tools: Gartner Research.

Kazman, R., Klein, M., & Clements, P. C. (2000). ATAM: Method for Architecture Evaluation. Carnegie Mellon University.

Kecman, V. (2001). Learning and Soft Computing: Support Vector Machines, Neural Networks, and Fuzzy Logic Models. MIT Press, Cambridge

Lankes, J. (2008). Metrics for Application Landscapes: Status Quo, Development, and a Case Study. Dissertation, Technische Universität München, München.

Lankes, J., & Schweda, C. M. (2008). Using Metrics to Evaluate Failure Propagation and Failure Impacts in Application Landscapes. In M. Bichler, T. Hess, H. Krcmar, U. Lechner, F. Matthes, A. Picot, et al. (Eds.), Multikonferenz Wirtschaftsinformatik, MKWI 2008, Proceedings. 1827–1838, GITO-Verlag, Berlin.

Lankhorst, M. M. (2005). Enterprise Architecture at Work: Modelling, communication, and analysis. Springer, Berlin

Peyret, H. (2007). The Forrester Wave: Enterprise Architecture Tools, Q2 2007.

Saritas, I., Etik, N., Allahverdi, N., & Sert, I. U. (2007). Fuzzy expert system design for operating room air-condition control systems. In B. Rachev (Ed.), Proceedings of the International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing, CompSysTech, ACM, New York

Technische Universität München, Chair for Informatics 19 (sebis) (2008). Enterprise Architecture Management Tool Survey 2008, from Technische Universität München, Chair for Informatics 19 (sebis).

Winter, R., & Fischer, R. (2006). Essential Layers, Artifacts, and Dependencies of Enterprise Architecture. EDOC Workshop on Trends in Enterprise Architecture Research (TEAR 2006), Hong Kong,

Wittenburg, A. (2007). Softwarekartographie: Modelle und Methoden zur systematischen Visualisierung von Anwendungslandschaften. Dissertation, Technische Universität München, München.

Zadeh, L. A. (1965). Fuzzy sets. Information and Control, 8(3), 338-353.