

## An approach to Discovery with miAamics and jABC

Christian Kubczak  
 Chair of Software Engineering  
 Universität Dortmund  
 christian.kubczak@cs.uni-dortmund.de

Christian Winkler  
 Chair of Service and Software Engineering  
 Universität Potsdam  
 winkler@cs.uni-potsdam.de

Tiziana Margaria  
 Chair of Service and Software Engineering  
 Universität Potsdam  
 margaria@cs.uni-potsdam.de

Bernhard Steffen  
 Chair of Programming Systems  
 Universität Dortmund  
 steffen@cs.uni-dortmund.de

### Abstract

We address the discovery scenario using miAamics, a framework for rule-based evaluation originally developed for efficient and scalable personalization purposes, as a reasoning engine. The discovery service is implemented in the jABC framework.

### 1 The miAamics Framework

The Challenge's discovery problem consists in dynamically identifying the best service provider among several shipment vendors to accomplish a given transportation task. Abstractly seen, miAamics<sup>1</sup> can be described as a rule based and situation aware matcher, that optimally matches profiles of requests to profiles of offers. It is situation aware in the sense that context information can also be taken into consideration in a context profile, which concurs in determining which offers fit and their ranking.

In the Challenge setting, an abstract description of a shipping request set in a certain context results in the suggestion of an *optimal vendor*, i.e. one which is able to fulfill this task and that optimizes secondary objectives like shipping costs or time. Fig. 1 sketches miAamics' domain modelling and its rule based evaluation components.

The *evaluation engine* uses formulas similar to event-condition-action rules to specify sets of business *rules* called *strategies*. The single request is then evaluated with respect to the currently enforced strategy. The evaluation is

<sup>1</sup>miAamics has been developed in 2000-2001 by METAFrame Technologies GmbH to address the needs of scalable real time personalization in CRM applications. The underlying technology is being patented. A later redesign of the framework in a Java environment has made it widely portable [3].

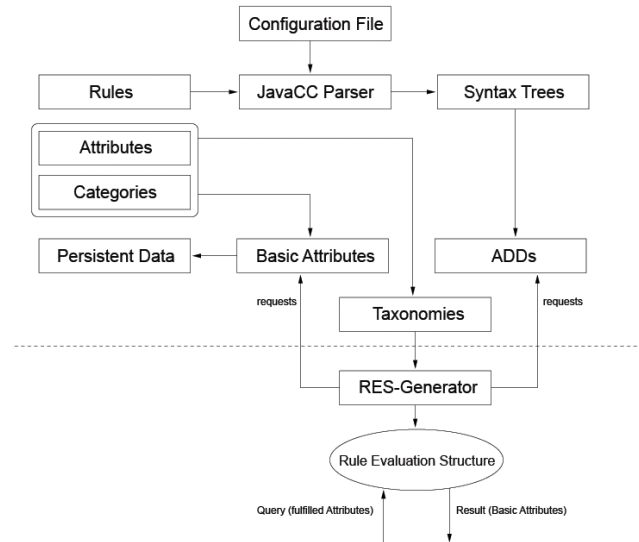


Figure 1. miAamics' domain modelling

global: all rules of a strategy are considered and the result yields the set of satisfying product offers (in this case the shipper services) ranked according to the optimality criteria expressed by the rules.

Since evaluation can be time expensive for a large offer basis, the evaluation can be also sped up by precomputing the effects of the evaluation in an optimized data structure (called RES), which is consulted at runtime, bypassing the case by case evaluation.

The *rules* do not refer directly to the entities involved (here only the shippers, in other applications the entities are much larger and heterogeneous catalogues of products) but to an *ontology*. We use here a simple ontology to express the

domain of discourse of the business (and therefore of the rules): a taxonomy based on abstract *attributes* that are meaningful for the business expert. For instance, shippers that take *heavy packages* are defined by an attribute *heavy packages* whereby the *maximum weight* data field of the shipper profile can exceed say 70 lbs. The corresponding definition of the attribute is *heavy packages*  $\geq 70$  lbs.

*Attributes* directly foot on the concrete profile vocabulary of the shippers, but rules are often more abstract, and refer to coarser concepts that combine different attributes. We call these coarser attributes *Categories*, which are conjunctions of different *Attributes*. Intuitively, there is an *is-a* relation between attributes and categories - that's why we say that our ontologies are in fact taxonomies.

miAamics' rules only make use of *Attributes* and *Categories*. As a consequence, if entities (like new shippers) are added dynamically to the application scenario after RES generation, the efficient evaluation structure is not touched. In fact, it directly delivers result sets that include the new items. Since RES structures are typically held in main memory, computing the answer of a query is extremely fast, allowing a real time response under very heavy load and thus the scalability needed for massive personalization as originally demanded in the CRM application.

Concrete offers (here the shippers' profiles and data) are stored in a backend database. Retrieving the information on the selected item(s) thus requires database access. Queries to the database use the constraints that define the taxonomy attributes. To optimize this last step to a single direct access we use in the RES *Basic Attributes*, representing the coarsest disjunct sets of data specified by attributes. The miAamics framework automatically computes the Basic Attributes, yielding a direct access to the data results.

As an efficient matching engine, miAamics brokers between input and output taxonomy-based profiles. It is thus an ideal backbone service for a special personalisation or evaluation task. The frontend, such as a graphical user interface to submit a query or display results, is independent of miAamics. It can for instance be reused from preexisting non-personalized versions of the application or website.

## 2 Rule Based Discovery

### Domain Model: Choosing the Vocabulary

We need first to fix a domain vocabulary, by analyzing the textual descriptions provided both in the scenario Wiki and in the WSDL files of the single shippers. We identify relevant data sets for inputs (the queries) and outputs (the shippers) and establish a domain-specific terminology at the miAamics' attribute level.

The shipping constraints are evinced too from those textual descriptions. For example, two constraints for the shipping vendors are:

```
Pack. weighing 50 lbs or less are ship.  
Collection is possible after 6.00
```

The corresponding data fields of a shipper look like this:

```
Max. package weight (lbs)  
Collection start time (24h)  
Collection end time (24h)
```

With this information we identify domain specific concepts to be specified in the taxonomy by *Attributes* and *Categories*. We discovered for example that it is important to characterize which shippers will ship lightweight packages. This is a relevant attribute in the taxonomy is defined by the constraint

```
Max. package weight (lbs) < 51
```

With this technique we identify and formalize the necessary information for gaining a complete classification of input and output data for the discovery. This ontology is provided in form of several text files at the Challenge's website.

The abstraction intrinsic with the use of taxonomies brings two limitations when dealing with concrete data values:

- miAamics's discrimination power is defined by the granularity level of the *Attributes*, and
- it cannot compute concrete values.

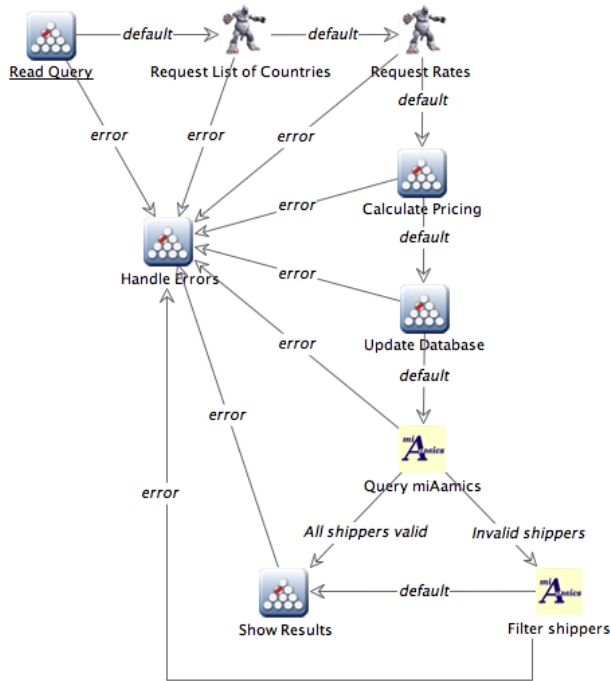
### Handling Discretization

The discrimination power is a discretization issue: it depends on the granularity of the attributes that relate to quantitative data fields. This is due to the desire to abstract and decouple from the persistent data. We increase discrimination power through finer granular attributes, e.g., instead of distinguishing only light, medium, and heavy packets, one can define attributes for any weight interval of interest.

In the Discovery scenario, we handle different day times for a shipping request by resolving a day interval into single hours, with one *Attribute* for each hour of the day. This granularity fits our needs since the described shippers only define relations to clock hours.

### Handling Dynamic Computations and Requests

For numerical computations we resort to an external pre-calculation mechanism, here used to deal for example with pricing rules that depend on package size and country rates. Since the *price* is a fixed data field included inside a shipper's profile, this information is computed and updated dynamically in the backend database. We use a simple pre-calculation and communication component that then forwards the now sufficiently defined shipping request to miAamics for evaluation. This application also takes care in general of invoking the web service of a shipper, to query



**Figure 2. The SWS Discovery SLG in the jABC**

missing data not provided by the textual description (e.g. destination countries, shipping rates, sometimes provided only on demand) and updates the shipper’s record accordingly at the backend of miAamics.

This is the typical usage pattern of miAamics, which is an embedded component. It serves as engine to efficiently evaluate given rule sets. It communicates with the Web or with the environment via an adequate a user interface component or a communication service that fits the special needs of the scenario (e.g. display results, data preparation).

### 3 The Discovery Application in the jABC

The full discovery solution consists of a jABC application, that includes miAamics as service that provides the matching technology. As already done for the Mediation [6, 4], we use the jABC environment [2, 5] to design a model driven discovery application that communicates with miAamics as embedded rule evaluation component.

Fig. 2 shows the jABC service logic graph of the discovery solution. Exactly as for the mediation, we compose graphically the business logic of the application by means of reusable SIBs (Service Independent Building blocks) that can be local or remote, components or services. The new SIBs constitute a new SIB library that extends the preexist-

ing library for the mediation component. Concretely,

- the ReadQuery SIB retrieves the information on a discovery request.
- since the textual descriptions of SWS Challenge do not statically provide all the necessary information for all shippers, we must retrieve the missing information dynamically for each request. A number of SIBs therefore query the vendor web services: Request List of Countries and Request Rates retrieve the list of countries a vendor ships to and the rates for those vendors that only support dynamic pricing, and Calculate Pricing computes the price for the given request taking into account the initial location, the destination, and special rates for domestic and international shipments.
- Now all prices for all the vendors are available and Update Database makes them available to miAamics.
- Query miAamics computes the set of fulfilling vendors, filtered Filter Shippers and ranked according to the preferences (cost functions) defined in the currently active strategy. For comparison of the effects we are defining two strategies, one that prefers fast and one that prefers pricely shippers.
- The ranking of the suitable shippers is then displayed (Show Results).

Every SIB has by construction a generic error branch. The error handling is here not further detailed. In a mature implementation of course it would.

As mentioned in Sect. 2 it is a matter of design and effort to decide how much to model in the miAamics ontology, to be evaluated with rules and policies, and what to deal with externally, via dedicated pre- or postprocessing SIBs that enrich the overall process. Depending on the complexity of the criteria (e.g. computing VAT and sale taxes, which exceed the expressiveness of the logic underlying the rules), the process option may be more advantageous, since it allows generic programs as filters.

From the joint jABC/miAamics point of view, it is important that we offer a variety of integrated possibilities. It is in fact well possible that established strategies may need to be adapted, but for fear of disruption specific filters that express only the desired difference of behaviour are quickly added as post-process SIBs for experimentation before integrating them as rules in the rule set.

We expect also the viceversa: criteria underlying rules and rule sets may need to be refined and become more complex, and eventually need to be extracted in a separate processing unit. For instance, when evolving from boolean decisions to complex classifications according to elaborate computations.

## 4 A miAamics User Session

We show here how to use the framework to concretely model the ontology and the rules for the Discovery scenario.

### Template based data collection

Since miAamics is independent of the data management layer, we use a flexible data management based on templates. Each datatype relevant to a personalisation scenario is a miAamics *Information Object* consisting of a set of datatype fields defined by *Information Object Templates*. Accordingly, the first thing to do when beginning to model a scenario within miAamics is defining IO-Templates.

As explained in Sec. 2 we define the domain-specific vocabulary for the shippers and requests by creating data fields (IO-templates) for all the information needed for the evaluation. Shipping rates, times, weights, dimension, etc. are among these. Afterwards the statically known persistent information for each entity (here, the various shippers) can be entered using data masks derived from those templates.

In this case we must set up everything manually from scratch. Usually, existing database schemas are imported automatically. For instance, the DbSchema plugin [9, 1] helps importing JDBC compliant database models in the jABC, avoiding effort and transcription errors.

### Taxonomies

As mentioned in Sec. 2, miAamics evaluates the matches on the basis of the Attributes and Categories of the domain taxonomy. For instance, the shipper's attribute *ships lightweight packages* is defined as *Max. package weight (lbs) lower than 51*, and foos on the database field (Information Object) *Max package weight (lbs)*. miAamics has a number of predefined relations for each field type, like *contains* for string fields.

Categories are graphically defined in a similar way, as conjunctions of attributes and other categories.

### Rules

The main issue when defining a personalisation schema is to identify and specify appropriate business rules fitting to the demands. The miAamics rule format is similar in flavour to enhanced Event-Condition-Action rules. Premisses and conclusions both range over Attributes and/or Categories and are arbitrary boolean expressions over taxonomy elements. Preferences are expressed by means of weights stating the significance of a rule: 'heavy' rules have a 'heavy' effect. The same rule can also appear with different weights.

Rules are then grouped in sets called *strategies*. Many strategies can be defined at the same time, but only one is active at any time. We defined two strategies, one preferring fast shipments and one preferring low priced ship-

ments. Some rules are different, but they also share some common rules and some that differ only in the weight.

### Result Evaluation Structure

Once a strategy is active, we can use it immediately fully dynamically, in interpreted mode.

To optimize the evaluation time, we can generate the Rule Evaluation Structure (RES) on one or more strategies. The RES structure is calculated and stored persistently in the backend database. The RES for the active structure is kept in main memory, yielding practically instantaneous answers to the queries.

## 5 Evaluation

Compared with the other approaches to the Discovery scenario, miAamics' strength is in the immediate response time once the RES has been computed, which makes sense whenever the service base and the rule base are fairly stable. Frequently changing service profiles and rule sets do not profit from this feature.

The current miAamics release is just the computational core. A previewer for rule evaluation, an import mechanisms for OWL ontologies e.g from Protégé [8]) and a link to reasoning tools such as Pellet [7] are in preparation.

## References

- [1] *DbSchema Website*, 2007. <http://jabac.cs.uni-dortmund.de:8002/plugins/dbschema/index.html>.
- [2] S. Jörges, C. Kubczak, R. Nagel, T. Margaria, and B. Steffen. Model-driven development with the jabac. In *Proc. HVC'06 IBM Haifa Verification Conference*. Springer Verlag, October 23-26 2006.
- [3] C. Kubczak. Entwicklung einer verteilten umgebung zur personalisierung von web-applikationen. Master's thesis, Universität Dortmund, March 2005.
- [4] C. Kubczak, B. Steffen, and T. Margaria. The jabac approach to mediation and choreography. In *2nd Semantic Web Service Challenge Workshop*, June 15-16 2006.
- [5] T. Margaria and B. Steffen. Service engineering: Linking business and it, cover feature. *IEEE Computing*, pages 53-63, October 2006.
- [6] T. Margaria, C. Winkler, C. Kubczak, B. Steffen, M. Brambilla, S. Ceri, D. Cerizza, E. D. Valle, F. Facca, and C. Tziviskou. The sws mediator with webml/webratio and jabac/jeti: A comparison. In *Proc. ICEIS'07, 9th Int. Conf. on Enterprise Information Systems*. to appear, June 2007.
- [7] *Pellet Website*, 2007. <http://pellet.owldl.com/>.
- [8] *Protégé Website*, 2007. <http://protege.stanford.edu/>.
- [9] C. Winkler. Entwicklung eines jabac-plugins zum design von jdbc-tauglichen datenbankschemata. Master's thesis, Universität Dortmund, March 2006.