Comparison: Handling Preferences with DIANE and miAamics

Ulrich Küster[1], Birgitta König-Ries[1], Tiziana Margaria[2], and Bernhard Steffen[3]

[1] Institute for Computer Science, University Jena, Germany,
{ukuester,koenig}@informatik.uni-jena.de
[2] Chair of Service and Software Engineering, University of Potsdam, Germany,
margaria@cs.uni-potsdam.de
[3] Chair of Programming Systems, Technical University of Dortmund, Germany,
steffen@cs.uni-dortmund.de

Summary. In this chapter we compare the DIANE and miAamics solutions to service discovery along a specific feature supported by those solutions: *preferences*. Although quite different in their theoretical and technical background, both techniques have in fact the ability to express user preferences, that are used internally to rank the evaluation results. These preferences are used here to incorporate functional aspects as defined by the SWS Challenge tasks, but they can also be used to express non-functional properties like quality aspects. Here we take a closer look at how preferences are realized in the two different approaches and we briefly compare their profiles.

1 Realizing Preferences with miAamics

The complete solution of the discovery scenario using as evaluation engine the miAamics personalization framework is described in detail in Chap. **??**. miAamics' weight mechanism can be used to define user preferences on the evaluation. This is the particular aspect central to the comparison with the DIANE solution.

Creating a domain model in miAamics is mainly a matter of creating sets of *attributes* that define profiles of customers, situations, and offers. miAamics can be seen as a situation aware matcher that matches profiles of customers to profiles of offers, taking into account profiles of situations where required. Evaluation goals are defined by *strategies*, i.e. sets of *rules* that use the aforementioned attributes. The miAamics evaluation engine calculates results concerning a given strategy by evaluating all the rules pertaining to this strategy and adding up the weights of the applicable rules.

miAamics' rules define mappings of the shape

$$Rulename: \quad f(request,situation) \rightarrow \{offer\}$$

hence the premises refer to customer and situation attributes while conclusions must refer only to attributes of offers. Rules are labelled with a name and a kind of numerical rating, in miAamics' terms *weight*. The following rules are examples for rules defined for the discovery scenario. Once again the premises regard to customer attributes that in the discovery scenario describe shipping requests, while the conclusions are defined by (shipment-) offer attributes.

1. Shipment to USA: Destination location is in USA → Ships to USA
2. Cheap shipment: Costs must be < $50 → Ships for less than $50
3. Express delivery: Has to be delivered within 24 hrs → Guarantees delivery within 24 hrs
4. Normal delivery: Has to be delivered within 48 hrs → Guarantees delivery within 48 hrs

Given a set of rules, it is now possible to define strategies. Strategies define which rules are considered for an evaluation and with which weight. If a rule is included in several strategies, it can be associated in different strategies with different numerical ratings, as shown in Table 1. The evaluation algorithm rates all offers by adding up the weights of the fulfilled rules. Thus the results can be ordered by their total weight, that reflects the quality of the solution for that strategy, as a sort of rank.

Following the scenario description, as a very simple example Strategy A in Table 1 only considers the first rule, *Shipment to USA*. In this case, the evaluation result for requests that have a USA destination address will only return shipment services that ship to the requested country (USA). As Rule 1 is the only rule considered, all the fulfilling offers have weight 100. Since they all fulfill the strategy's constraints equally well, but a single offer is at the end selected, this happens by random choice among them.

Rule No.	Strategy A	Strategy B	Strategy C
1	100	100	100
2	*(not used)*	40	50
3	*(not used)*	25	20
4	*(not used)*	25	20

Table 1. Different rules and weights in different strategies

Functional and non-functional preferences using weights

In the more realistic scenarios, a customer specifies several quality criteria in his requests. (Situated) quality criteria are expressed in miAamics via rules, so more refined strategies include several rules. The rule base described above

considers also criteria as the shipping price and time for delivery, besides the destination country. Both Strategy B and C in Table 1 include all four rules, but with different weights. Taking a look at the weights defined for the rules in a single strategy, it is possible to describe in detail and intuitively how this strategy sets the user preferences for the request evaluation. In both strategies, the functional aspect is covered by Rule 1: that is the essential criterion to determine if a shipper is eligible. The other rules express preference criteria.

Strategy B ensures that a delivery to the requested country is possible because the value for this Rule (100) cannot be surpassed by the total weight of other rules (90). So, no shipper is ever going to be selected that does not ship to the USA - the central selection criterium. Furthermore, Strategy B privileges offers that can guarantee a delivery within 24 hours over cheap ones. If a shipment within 24 hours is offered (Rule 3), Rule 4 is also fulfilled, since a shipment within 24 hours is also within 48 hours. Thus a sum of 50 points is achieved in this case while cheap offers can only gain 40 additional points. Cheap and fast shippers can get the sum of the points, 90.

For a different customer profile preferences can be set in a different way via a different strategy, using for example the same rules but with adjusted weights. For example, if it is more important to chose the cheapest offer than to deliver a package within short time, weights could be defined as shown in Strategy C. Once again, delivery to the selected country is ensured, but this time cheap offers are preferred (50). Delivery time is still considered: cheap and quick shippers will be preferred (become a higher rating) over those that are simply cheap.

This example covers both functional and nonfunctional criteria. The design of the weights is sensitive, since it determines the relative relevance of the rules (and the criteria they express) in the overall selection strategy.

2 Realizing Preferences with DIANE

DIANE uses a set-based modeling approach for service discovery. Service providers usually do not offer a single service instance, but are able to provide a family of similar services, e.g. shipping to a multitude of locations. Thus, they describe their offer as the set of services they can potentially provide. The shipping services of the SWS-Challenge, for instance, provide shipping of packages that adhere to certain weight and size restrictions within given sets of countries.

Requesters with a certain need, on the other hand, may have a perfect service in mind, but usually accept quite different services based on the available offers. While a fast, reliable, and cheap shipping offer would obviously be a perfect choice, in reality a fast but expensive shipping offer might be as acceptable as a slower but also less expensive one. Therefore, service requests in DIANE describe the set of service instances which are acceptable to the requester. In order to express preferences among services, requesters may use

fuzzy instead of crisp sets. A higher degree of membership in the fuzzy request sets corresponds to a higher preference for that particular service. For further information about service modeling and how services are matched in DIANE we refer to Chapter **??**. In the following we detail how user preferences are expressed in the DIANE context by the examples of the SWS-Challenge discovery scenarios.

2.1 Basic Preferences

Within the SWS-Challenge, preferences were mainly used in the hardware purchasing scenario. Goal B1 for instance requires to purchase a notebook with certain properties (e.g. at least 60 GB harddisc size) and states that price matters most, thus cheaper offers should be preferred if the other requirements are met. Such basic preferences are captured by *direct conditions* on request attributes, in this case on the price of the product. A crisp condition like "`<= 2000`" on the price attribute of a product requires the price of that product to be less than 2000 but does not encode preference for lower prices. These are expressed using fuzzy conditions like "`~==[0, 2000] 0`". This expression requests the price to be zero, but accepts values from the interval $[0, 2000]$ with linearly decreasing preference. Similarly an expression like "`~== 2007-11-11T11:11`" on the pickup time asks for pickup at the specified time, but accepts a default deviation of up to three days – again, with linearly decreasing preference.

2.2 Advanced Preferences

Fuzzy direct conditions are sufficient to express preferences on single attributes. However, most realistic scenarios involve preferences on multiple attributes with conflicting optimization goals, like preferences for a low price and a high-quality configuration at the same time. In such cases the different goals have to be balanced according to the user's preferences. In DIANE, this is achieved with *connecting strategies*. During the matchmaking, complex types are compared by recursively comparing their attributes and combining the retrieved values. The matchvalue of a particular notebook with respect to a notebook purchasing request, for instance, is obtained by combining the matchvalues obtained from comparing the attributes of the notebook (price, HDD size, RAM size, processor type, ...) with the corresponding requirements from the request. The resulting values are normalized to the interval $[0, 1]$ and by default combined by multiplying them. To emphasize particular conditions compared to others, a requester specifies a custom connection strategy, basically a function which maps the set of attribute matchvalues to the interval $[0, 1]$. Available base functions are product, weighted sum, min and max as well as the exponential function. Goal C4, for instance, specifies that the processor power of the notebook is most important and that more RAM is more important than a bigger hard disc. This is captured by the

connecting strategy $processor^3 \times memory^2 \times hardDisc$ which penalizes lower values for the processor power most, and lower values for the memory size more than those for the hard disc size.

2.3 Practicalities

The definition of appropriate preferences is not always trivial (see Section 3) and requires a certain knowledge about DSD. DIANE thus supports the separation of concerns via parameterized *request templates*. Any concept in a DSD request can be replaced by a request input variable, thus parameterizing the request. This way, trained domain experts can create one or more DSD request corresponding to different preference profiles. In the shipping example, for instance, different request templates might correspond to requests prioritizing fast shipment, inexpensive shipment or precisely matching pickup times. End users can then select the most appropriate request template according to their needs and use them by simply providing values for the required input variables, i.e. the cargo to be shipped and the locations for the pickup and the delivery. This way, technical details of DSD can be hidden behind simple user interfaces, making dealing with DSD directly unnecessary for end users.

3 Challenging Example: Goal B2

Balancing preference criteria with different nature is not always easy in both approaches. Despite the fact that most preferences could be encoded very intuitively, one goal posed difficulties. Goal B2 of the second scenario prefers black notebooks compared to white ones, but prefers to buy the white one if it is significantly (more than $100) less expensive than the black one. Such a rule-based preference, combining a constant part (the color) and a variable part (price difference) does not map well neither to the fuzzy set-based preference mechanism of DIANE nor to miAamics.

Since the price limit was set to $1800 a price difference of $100 results in a difference of the matchvalue in DIANE of roughly 0.055 (100/1800). Therefore the preference values for the color were chosen to reflect exactly this difference: black [1] and white [0,944]. A weighted sum of $0.5 \times price + 0.5 \times color$ however does not result in the desired behavior. The problem is that any black notebook regardless of the price or any product whatsoever that does not cost much would result in a matchvalue of at least 0.5. Therefore the connecting strategy was extended to $min(color, price, 0.5 \times price + 0.5 \times color)$. However, this way the matchvalue was soon dominated by the matchvalue for the price and the weighted sum did not influence the outcome any more. To resolve this issue, the influence of the weighted sum was strengthened using a polynome yielding: $min(color, price, (0.5 \times price + 0.5 \times color)^5)$. This way the desired behavior could be successfully achieved.

Using miAamics, just expressing the goal is not a big problem. Defining price rules for each $100 interval with a weight of e.g. 50 and an additional rule for black devices with a weight that is less than the 50 points of the price rules would roughly lead to the desired behavior. But this example also shows the drawbacks of the miAamics solution. Since miAamics is based on boolean rules, conditions on numerical values, e.g. the price, need to be discretized. This can lead to a huge amount of rules. Moreover, within a discretization interval for a criterion (here price), offers are indistinguishable for that criterion. As a consequence, two black notebooks priced $900 and $999 could not be differentiated in this example, while the DIANE solution would prefer the cheaper one. Furthermore the behavior is as desired only if the price of a black notebook is close to the upper bound of a price interval. For example a white notebook for $899 would beat a black one for $900 though it is not *significantly* cheaper. This is due to the fact that the white notebook falls in another price interval than the black one. This problem can only be reduced by choosing smaller price intervals and thus creating more rules - which can happen using our automatic rule generation facilities.

4 Comparison

The discovery approaches are different in four dimensions:

1. selection mechanism (underlying technology)
2. mode of use (pragmatics)
3. profile of users (who can do what when)
4. performance

4.1 Selection mechanism (underlying technology)

DIANE builds on arithmetics. In particular, it allows one to automatically weight and compare numerical parameters, and to specify a prioritized selection of data/products/offers based on those comparisons. E.g., price differences may be transformed into preference values, which may then be put into the context of other preference values, e.g. for timeliness, quality assurance, or color in a preference-based fashion. This is typically done by building a product of powers of the involved preference values: the higher the power of a value the greater its impact. This method also allows one to indirectly code a priority scheme between preference values if desired.

In contrast, miAamics is based on predicate logics: simple if-then rules describe single *aspects* of when a certain data/product/offer fits. The overall selection is then based on large sets of such rules, each weighted according to the relevance/significance of the modeled aspect. If various rules propose the same data/product/offer, the weights for these data/products/offers are added, and the data/products/offers with the highest such sums are the winners. This way of aspect-oriented modeling is highly compositional: to add a

new aspect, one simply needs to add the according rules, weighted reflecting the according relevance/significance. Preference can be modeled indirectly using the weights, e.g. giving the pricing aspect a higher weight than the sum of the weights of competing aspects. Numerical values can be treated based on adequate discretization.

4.2 Mode of use (pragmatics)

Both approaches are based on ontologies/taxonomies, but their mode of operation is quite different:

DIANE is based on complex ad hoc queries, which describe the overall pattern of selection (see above): a user describes his desires and preferences according to DIANE's selection mechanism in a monolithic fashion. The resulting expression needs to be fully evaluated at runtime according to the current state of the environment (data bases etc.).

In contrast, miAamics' specification of selection is decomposed in two parts:

- the specification of the set of weighted rules describing the aspect-oriented relevance of data/products/offers. In our experience, these rules can be dealt with be business experts without IT knowledge after a short training.
- predicates describing the individual preference of a user for a certain selection process. These predicates may describe the user's price, quality, or color sensitivity, which may occur in the rules' 'if' part, thus steering which of the rules defined in in the first part are fired and which not. Setting these predicates, which means setting the profile of selection, can easily be done simply by clicking at certain preferences. This is so intuitive that it does not even require an explanation.

4.3 Profile of users (who can do what when?)

The correct conceptualization of a problem domain and the design of strategies or queries that appropriately capture an end user's preferences and desires is often a non-trivial task. Therefore, both approaches allow to distinguish two profiles of users:

- The *domain expert* conceptualizes the problem domain and develops appropriate parameterized query templates (in case of DIANE) respectively an aspect-oriented weighted rule scheme together with a set of strategies (in case of miAmics).
- The *end user* only selects and customizes the request by choosing appropriate parameter values for one of the predefined query templates (DIANE) respectivly by selecting some predicates for preference (miAmics).

In particular the latter role is open to a very wide public. Almost all internet users will be able to perform this kind of selection without requiring even an

explanation. But also the first role does not require extensive IT knowledge. Both approaches envision domain experts with some basic training in the employed technology as the optimal clientele for the task. In some cases, in particular when the preferences are inherently complex, DSD might require a better mathematical understanding from the user than miAmics.

4.4 Performance

The runtime complexity of evaluating a DIANE request with respect to a single offer is roughly linear in the size of the request (i.e. the number of attributes). Thus, large requests together with large numbers of offers may pose a performance problem in a context, where thousands of users operate in parallel.

In contrast, miAamics technology comes with a compilation process, which transforms large sets of weighted rules into simple evaluation structures. This guarantees an extremely fast selection (orders of magnitude faster than the DIANE selection), and therefore scalability.

4.5 Summary

Overall, miAamics and DIANE were designed with very different goals: miAamics was designed with a focus on ease of use and scalability whereas DIANE was designed with a focus on expressivity and precision. This corresponds to the fact that miAamics is based on boolean rules and DIANE on arithmetics and fuzzy set theory. Consequently, DIANE is more flexible for specification, in particular considering the treatment of continous numerical values, that needs to be discretized in miAamics. On the other hand, miAamics restricted expressivity makes the specification of rules very easy and allows to achieve superior runtime performance. Thus the two technologies have complementary profiles, and therefore their own right of existence.