Implementation of Similarity Measures for Event Sequences in *myCBR*

Eduardo Lupiani[1], Christian Sauer[2], Thomas Roth-Berghofer[2], Jose M. Juarez[1], and Jose Palma[1]

[1] University of Murcia, Spain
{elupiani,jmjuarez,jtpalma}@um.es
[2] The University of West London, UK
{Thomas.Roth-Berghofer,Christian.Sauer}@uwl.ac.uk

Abstract. The computation of the similarities between event sequences is important for many fields because many activities follow a sequential order. For instance, consider an industrial plant that triggers different types of alarm due to detected event sequences or the treatment sequence that a patient receives while he/she is hospitalized. With the appropriate tools and techniques to compute the similarity between two event sequences we may be able to detect patterns or regularities in event data and so be able to perform predictions or recommendations based on detected similar sequences. This work describes the implementation of two event sequence similarity measures in *myCBR*, with the purpose of creating a similarity measurement approach for complex domains that employs the use of event sequences. Finally, initial experiments are performed in order to study if the proposed measures and measurement approach are able to predict future situations based on similar event sequences.

1 Introduction

Using event sequences is practical for many scenarios, particularly scenarios that aim to detect patterns in the occurrence of events or to retrieve past event sequences that are similar to a current one. Examples of event-based data include medical records and procedures, data in a temporal context such as historical, biographical and career path data, internet session data, traffic incident data, process control data, and administrative process data [12].

In the above mentioned scenarios Case-based Reasoning (henceforth CBR), which is a problem-solving methodology [1,13], can play an important role. For instance, assume the problem of an industrial plant being monitored to supervise its proper functionality. With a CBR system using event sequence similarity measures representing a sequence of measurements or alarms (from sensors in the plant), it is possible to retrieve previous sequences of measurements/alarms that indicated an imminent failure of the processing plant. If these sequences can be detected early enough, then it could be possible to set up a maintenance task to fix the problem, so the plant can stay operational.

Within the CBR community there already exists other work on how to make use of temporal representations, such as similarity between workflows using an edit distance [6], graph matching to supervise business workflows [3] and to find software for reusing [14], and even the use of episodes (sequence of cases) in order to supervise a wastewater treatment process [9]. However, while there are approaches to compute the similarity between two event sequences [2,7,8,11,12,15], only very few or almost none of these approaches were ever implemented in a software, which in turn is making it difficult to select the most suitable event sequence similarity measure for a specific software system.

For this reason our work presented in this paper focuses on the implementation of similarity measures between event sequences, so an actual implemented CBR system can be employed in the problem domains described before. For this purpose we are extending the *myCBR* development software to implement the event sequence similarity measures. The main reason for our choice is that the *myCBR* software is intended for building prototypical CBR applications in teaching, research, and small industrial projects with low effort [10]. Furthermore, it is designed for being easily extended with new functionalities.

After the implementations of the new event sequence similarity measures we also were interested in studying whether they were able to detect similar event sequences in a proper way. We were especially interested in the question if it was possible to build a CBR system able to predict future situations based on the newly implemented recognition of similar event sequences. To study these aspects we have conducted initial experiments with a case-base consisting of event sequences that represent risky situations in an industrial furnace.

The remainder of this work is as follows: in the next section we review the background of this work. We review the definitions of event sequences and similarity between events, as well as details of two similarity measures. In this section we also describe the *myCBR* software. In section 3 we describe in detail how the similarity measures have been implemented in *myCBR*. Section 4 describes our initial experiments performed using an artificial case-base with cases that contain event sequences. Finally, in section 5 we present our conclusions and describe planned future work.

2 Background

Previous to any explanation of similarities between event sequences, it is necessary to define what is an *event*, *event type* and an *event sequence*.

2.1 Event Sequences

The following definition of event is based on the definitions given in [7], [11] and [8], which give similar explanations of what an event is and are unified here into one definition that encompasses them all.

Definition 1. *An event is a tuple consisting of a label representing an action or activity and a time-stamp that represents the moment when the action occurs [7,11]. Formally we can define an event e as following:*

$$e = (a, t)\,,$$

with $e \in \mathbb{E}$, being $\mathbb{E}$ the domain of all the possible events, a as the taken action or activity, and $t \in \mathbb{R}$ as the time-stamp of the event.

The activity a can be represented as a simple value of the following data-types: integer, string or symbol. However, complex representations are possible as well, by defining event types [8]. The event type determines the data structure of each activity in the problem domain and this does not affect to the time-stamp of the event. Furthermore the amount of the set of event types to define is determined by the domain of the problem to resolve [7], so the values of this set relies on the application itself.

Definition 2. *An event sequence is a partially ordered set of events:*

$$S = \langle (e_1, t_1), (e_2, t_2), \ldots, (e_n, t_n) \rangle$$

where e_i are the possible event types in the problem domain, and it holds that $t_i \leq t_{i+1} \forall i = 1, \ldots, n-1$ [7].

Example 1. Assume a processing plant that may trigger four types of alarms from sensor readings of temperatures LL, L, H, HH. From the definition given for event sequence it is possible to represent a sequence e of alarms as follows:

$$e = \langle (L, 1), (H, 2), (HH, 3), (HH, 5), (LL, 6) \rangle$$

2.2 Similarity Measures for Event Sequences

There are different ways to compute the similarity between two event sequences. When two event sequences have the same number of events it is possible to use a *lock-step* measure [16]. The basis of this type of measure is to compare the i-th event in both sequences and establish the (local) similarity of this event pair and at the end to compute a global similarity measure of the two sequences based on their local similarities of event pairs. However this similarity measure is not suitable when the length of the two sequences differ, in this case it is convenient to employ other types of similarity measures. The Edit Distance measures the cost of transforming one event sequence into another. Therefore, the similarity between the two sequences, which are called pattern and query, is given by the number of used transformation operations, i.e., the lower the number of needed transformation operations to transform the query sequence into the pattern, then the higher the similarity between the two event sequences is.

Prior to showing the event sequence distance measures, we propose the following definition of distance and similarity. Given the space of possible events $\mathbb{E}$, the distance function between two event sequences is defined as $d : \mathbb{E} \times \mathbb{E} \to \mathbb{R}$.

Additionally, in some cases it is useful to normalise the distance between two event sequences in the interval $[0, 1]$, thus $d : \mathbb{E} \times \mathbb{E} \to [0, 1]$. This is possible by dividing the resulting distance between two sequences by the maximum observed value. When the distance function is normalised, defining the similarity between two event sequences e^x, e^y is possible as follows:

$$\begin{aligned} sim :& \mathbb{E} \times \mathbb{E} \to [0, 1] \\ sim(e^x, e^y) =& 1 - d(e^x, e^y) \end{aligned}$$

The following subsections are based on the state of the art for computing the similarity between two event sequences.

Edit distance between two Sequences The *edit distance* measures the related cost of transforming an event sequence into another, using operations of insertion, deletion or alignment/movement. This measure is also known as *Levenshtein distance* because of the last name of the author who proposed it [4]. Although the Levenshtein distance measure only involves strings. In the works of [5,7] it is proposed to use a modification of the edit distance to manage event sequences, where the alignment/movement involves the temporal distance between two events. This algorithm requires that given two events of the same type, the distance between their timestamps must be lower than the cost of any other operation. If this requirement is not met the cost of aligning two events with the same event type can be higher than the inserting and removing operations. Formally, it can be defined as following:

Definition 3. *Let* $S_x = \langle (e_1^x, t_1^x), \ldots, (e_n^x, t_n^x) \rangle$ *and* $S_y = \langle (e_1^y, t_1^y), \ldots, (e_m^y, t_m^y) \rangle$ *be two even type sequences. The edit distance between* S_x *and* S_y *is the number of operations to transform the sequence* S_x *into* S_y *[7].*

So to normalise the edit distances in the interval $[0, 1]$, it is necessary to divide the resulting distance by the maximum necessary effort to transform one sequence into the other, where the maximum effort is the maximum number of operations needed to build the longest event sequence from scratch.

Match & Mismatch Similarity [16,15] The Match & Mismatch (henceforth M&M) is a measure based on the idea of edit distance, thus it measures the cost of transforming one event sequence into another. This measure splits each event sequence into several event sub-sequences, sorting them into lists for each event type [16]. Using this approach the algorithm can compute the distance between sequences that contain the same event type, which is a simpler sub problem [15] than the initial problem.

For each event type the algorithm computes an edit-distance matrix. From all the built matrices, four different measures are calculated: the time difference between the events (TD), the missing events (NM), the extra events (NM) and the number of swamping events (SN). These measures are the basis for an overall score that measures the similarity between the two event sequences.

Unlike the Edit Distance, which is a distance, the M & M returns a similarity score in the interval $[0.01, 0.99]$.

2.3 *myCBR* Software Development Kit

The key motivation for implementing and further developing the open source software *myCBR* was and still is the need for a compact and easy-to-use tool for building prototypical CBR applications. The use of the *myCBR* tool especially aims for the sectors of teaching, research and small industrial projects with low initial development effort. To allow for the rapid prototyping of CBR systems, *myCBR Workbench* provides graphical user interfaces for modelling attribute-specific similarity measures and for evaluating the resulting retrieval quality in an integrated retrieval interface. In order to further reduce the development effort of CBR systems, especially the built up of a case-base, after defining an appropriate case representation, *myCBR Workbench* includes tools for generating the case representation automatically from existing raw data as well as importing cases from CSV data files. The accompanying Software Development Kit (SDK) allows for the integration of the developed CBR systems into other applications and allows also for the implementation of extensions into the SDK based on specific requirements such as additional similarity calculations, as it is demonstrated in the present research work.

2.4 Building structural knowledge models with *myCBR Workbench*

myCBR provides a workbench, the *myCBR Workbench*, that supports the creation and maintenance of the vocabulary and similarity measures used within a CBR system and thus provides control over the knowledge model of a CBR system developed or maintained using *myCBR*. The *myCBR Workbench* is implemented as using the *Rich Client Platform* (RCP) of the *Eclipse* JDE. *myCBR Workbench* offers two different perspectives on a CBR system being developed - a perspective for the knowledge modeling and another perspective to develop and maintain the case-base(s). Figure 1 provides an overview of the modeling perspective within the *myCBR Workbench*. The conceptual idea behind the modeling perspective is that a case structure is created first being followed by the definition of the vocabulary and the creation of individual local similarity measures for each attribute. A description of a *concept* can contain one or more attribute descriptions as well as references to other concepts, hence allowing the user to create object-oriented like case representations. A description of an *attribute* can be formulated using one of the following data types: Boolean, Double, Integer, Interval, String or Symbol. For each data type *myCBR* provides a default similarity function that supports their definition and individual functionalities to further define more sophisticated similarity functions with regard to the specific data types chosen for an attribute. To ease the prototyping of CBR systems one single attribute description can have more than one similarity measure, allowing for rapid testing and experimentation to find the most suitable similarity functions.

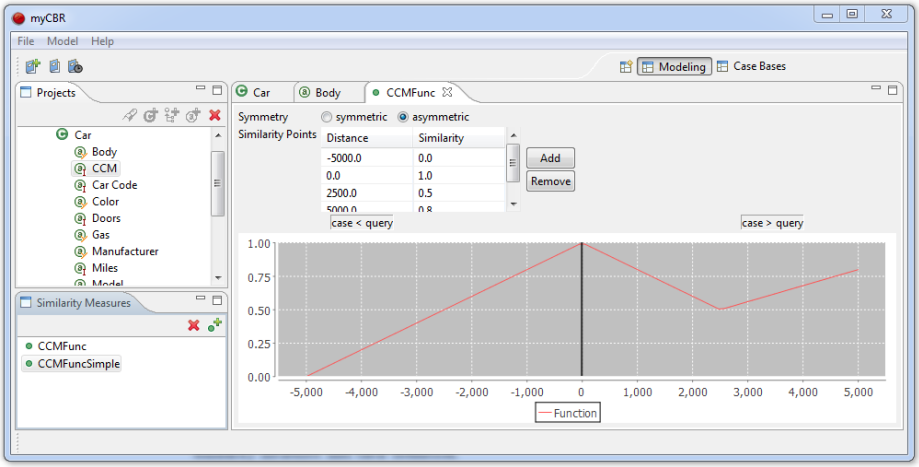

Fig. 1. The modeling perspective of *myCBR Workbench* with the case structure (left), a list of available similarity measures (left bottom) and a view of their structure (center)

As already described the modelling of the similarity measures in *myCBR Workbench* takes place first on the attribute level, defining the local similarity measures and then moves on to the concept level to define the global similarity measure(s) of a case representation. The attributes are then defined with their data types and value ranges. Depending on the chosen data type of an attribute the *myCBR Workbench* provides modelling GUI's for: Numerical data, providing predefined distance functions along with predefined similarity behaviour (constant, single step or polynomial similarity in-/de-crease). For the definition of similarity measures for symbolic values, *myCBR Workbench* provides table functions and taxonomy functions. A table function allows defining individual similarity values for each value pair, while a taxonomy subsumes similarity values for subsets of values. Depending on the size of a vocabulary, table similarity measures are hard to maintain and taxonomies provide a more structured overview. For symbolic values, also set similarities are provided in order to compare multiple value pairs [10].

The present paper describes how *myCBR* can be extended in order to include a new data type such as event sequences, as well as two distances to compute the similarity between event sequences. Therefore the highly modularised code structure of the *myCBR* project was extended to allow for the implementation of the new attribute and the computation of similarities for it's new data type being event sequences.

3 Implementation of the Similarity Measures in *myCBR*

The major goal of *myCBR* is to minimize the effort for building CBR applications that require knowledge-intensive similarity measures[10], so the structure

and design of the *myCBR* software is ready for being extended with new functionalities. The software is implemented in Java, so it follows an object oriented approach. In this work we are only showing the set of classes that are related with the representation of the case-base and the processes to retrieve cases. The figure 2 depicts the class diagram of the classes realising the retrieval process and the case representation in a case-base.

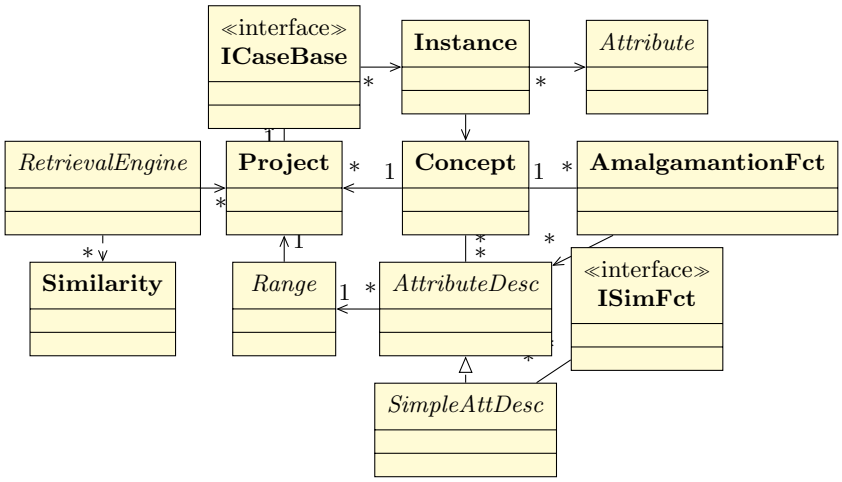

Fig. 2. Main classes of the *myCBR* SDK related to the similarity computation process.

Where each class has its own role:

Project: A Project consists of a concept that defines the vocabulary of a project domain and one or more case bases storing problem cases from the domain.

Instance: Represents values occurring in either query and cases. It contains a set of attributes that conforms to the case structure.

Attribute: This is an interface with the basic operations of all types of attributes. Each attribute must have an implementation of this class. For instance, DoubleAttribute, StringAttribute, etc.

ICaseBase: An interface that represents the set of retained cases.

RetrievalEngine: Implementation of the process to retrieve the most similar cases to a query.

Concept: Vocabulary of the problem domain.

Range: Possible values of a particular implementation of an attribute.

AttributeDesc: Contains details about the description of one attribute of the case. For instance, there are descriptions for numbers, symbols, dates, etc.

SimpleAttDesc: This class extends the AttributeDesc class with new fields and methods for being used by a ISimFct.

ISimFct: Implementation of the similarity between two attributes with the same description, i.e. between attributes of the same data-type.

AmalgamationFct: Implementation of the global similarity function to compute the similarity between two instances / cases, considering all the existing ISimFct to create a Similarity object.

Similarity: The value that represent the similarity between two instances.

Our approach was the creation of a new attribute that represents event sequences. From the definitions given before in section 2, we have set the following string representation:

$$sec =< (concept, timestamp) \ldots >$$

The figure 3 depicts the new classes that we have implemented. With them we can represent an event sequence within as an attribute of a case which allows myCBR to compute the similarity between two different event sequences.

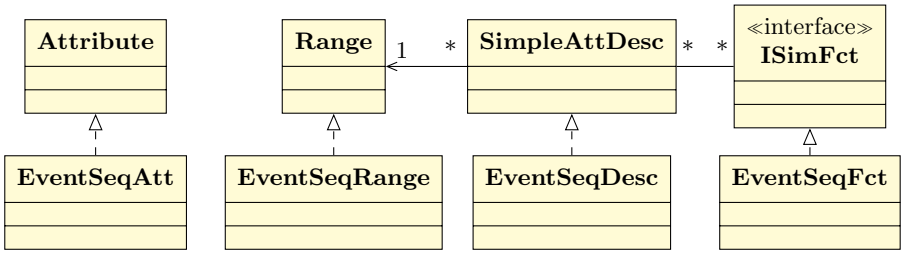

Fig. 3. Sequence of operations to compute the similarity between two cases.

EventSeqAtt: Contains the information of a particular event sequence.

EventSeqRange: Contains the possible values of event sequences. It is responsible of creating new *EventSeqAtt*.

EventSeqDesc: Contains the details about how the event sequences are implemented.

EventSeqFct: This corresponds to the implementation of the similarity measures that we have worked on in the present paper. When an object of this class is created it is mandatory to specify which type of measure it should employ.

The implementations of the measures are expressed in algorithms 1 and 2. The algorithms are in pseudo-code to ease their comprehension; however they are implemented using Java.

Algorithm 1 Calculate the edit distance $d_S(S_x, S_y)$ between two sequences S_x, S_y

Input: Two event sequences $S_x = \langle (e_1^x, t_1^x), \ldots, (e_n^x, t_n^x) \rangle$ and $S_y = \langle (e_1^y, t_1^y), \ldots, (e_m^y, t_m^y) \rangle$, with $e^x, e^y \in \mathbb{E}$, the costs $w(e^x), w(e^y)$ of the *insertion* and *deletion* operations.
Output: Edit distance $d_S(S_x, S_y)$ between the two given sequences.

$r \leftarrow$ matrix of $n \times m$ dimensions
$r(0,0) \leftarrow 0$
for $i \leftarrow 0$ **to** m **do**
  $r(i,0) \leftarrow r(i-1,0) + w(e^x)$
end for
for $j \leftarrow 0$ **to** n **do**
  $r(0,j) \leftarrow r(0,j-1) + w(e^y)$
end for
for $i \leftarrow 1$ **to** m **do**
  for $j \leftarrow 1$ **to** n **do**
    $upd_{S_x} \leftarrow r(i-1,j) + w(e^x)$
    $upd_{S_y} \leftarrow r(i,j-1) + w(e^y)$
    $align \leftarrow r(i-1,j-1)$
    if $e^x = e^y$ **then**
      $align \leftarrow align + (0.5 \times |t_i^x - t_j^y|)$
    else
      $align \leftarrow align + w(e^x) + w(e^y)$
    end if
    $r(i,j) \leftarrow min\left(upd_{S_x}, upd_{S_y}, align\right)$
  end for
end for
return $r(n,m)$

Algorithm 2 Calculate the M&M distance $MM(S_x, S_y)$ between two sequences S_x, S_y

Input: Two event sequences $S_x = \langle (e_1, t_1^x), \ldots, (e_n, t_n^x) \rangle$ and $S_y = \langle (e_1, t_1^y), \ldots, (e_m, t_m^y) \rangle$.
Output: Match & Mismatch similarity between the two given sequences.

$TD \leftarrow 0, NM \leftarrow 0, NE \leftarrow 0, NS \leftarrow 0$
for $e \in T_\epsilon$ **do**
  $sec_x^e \leftarrow$ the set of events of type e in S_x
  $sec_y^e \leftarrow$ the set of events of type e in S_y
  if $|sec_x^e| \leq |sec_y^e|$ **then**
    $NE \leftarrow NE + |sec_x^e| - |sec_y^e|$
    $aux \leftarrow sec_x^e$
    $sec_x^e \leftarrow sec_y^e$
    $sec_y^e \leftarrow aux$
  else
    $NM \leftarrow NM + |sec_x^e| - |sec_y^e|$
  end if
  $n \leftarrow |sec_x^e|, m \leftarrow |sec_y^e|, diff \leftarrow n - m$
  $c \leftarrow$ matrix of $n \times m$
  $c(0,0) \leftarrow 0$
  for $j \leftarrow 0$ **to** $m-1$ **do**
    for $i \leftarrow 0$ **to** $diff$ **do**
      $cost \leftarrow |t_{j+i}^x - t_j^y|$
      if $j > 0$ **then**
        $cost \leftarrow cost + c(i, j-1)$
      end if
      if $i > 0$ **then**
        $c(i,j) \leftarrow min(cost, c(i-1, j))$
      else
        $c(i,j) \leftarrow cost$
      end if
    end for
  end for
  $TD \leftarrow TD + c(difference, m-2)$
end for
$NS \leftarrow$ number of swamping events
return $4 - w_{TD}TD - w_{nm}NM - w_{ne}NE - w_{ns}NS$

4 Initial Experiments

For our experiments we have created a synthetic case-base that represents the knowledge related to the operation of an industrial furnace. The purpose of the evaluation was to analyse whether the CBR system was able to predict unsafe situations from cases that represent the operation of the furnace. Thus, the cases consisted of event sequences describing the event data of a whole day of furnace operation and a label to classify the event sequence as being "normal" or "not normal" with regard to the final safety situation of the event sequence. Each event sequence had up to four different event types: LL, L, H and HH, that represented the temperature of the furnace, where the event type HH meant an unsafe situation that must be avoided. So as to graphically represent the structure of every case, the figure 4 depicts three sample cases: two cases describing normal operations and one describing an unsafe situation (not normal).

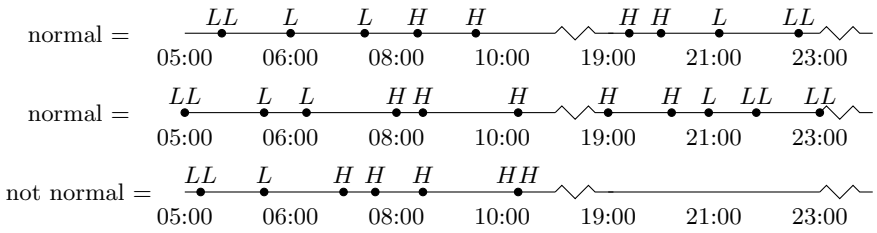

Fig. 4. Examples of three event sequences

The case-base contained 1000 cases, where each case contained an event sequence of one day duration. In order to analyse if the system predicted unsafe situations correctly, we built a test set with different event sequences that may appear during the operation of the furnace within a window of time. The size of this window or in other word the duration of the 'partial' query sequence was always smaller than the entire, one full day, event sequences covered in the cases.

The modified *myCBR* system which we experimented with uses a K-Nearest Neighbours approach to retrieve the most similar case to an input case. As adaptation mechanism, the most common solution between the nearest neighbours is the returned solution to the input event sequence. The system is evaluated using a Cross-Validation approach, with the number of neighbours (k) within the set $\{1, 3, 5, 7, 9\}$. The values that the evaluation computes are the error rate, the false positives of "non normal" event sequences and the number of times that the CBR system predicted unsafe situations in advance. Detecting unsafe situations in advance meant that the retrieved case contained the event sequence of a "non normal" or unsafe situation and was detected (retrieved) before the risky HH event arises.

	Edit	M&M	Edit	M&M	Edit	M&M
Neighbours	**Error rate**		**False Positives**		**Predicted**	
1	0.3200	0.2060	**0.0145**	0.0290	**0.7536**	0.7391
3	0.0220	0.0190	0.0435	0.0290	0.5942	0.5942
5	**0.0020**	0.0040	0.0290	0.0580	0.5942	0.5942
7	0.0030	0.0040	0.0435	0.0580	0.5942	0.5942
9	0.0060	0.0060	0.0870	0.0870	0.5942	0.5942

Table 1. Results of the experiments with the Edit Distance and M&M measures. The best results are highlighted in bold.

The best results for each measure are highlighted in bold. Regarding the results, it seems that for this particular experiment the Edit Distance achieved better results. Besides having only one neighbour results into the worst error rate for both measures, in this configuration the number of false positives and the number of predicted "not normal" situations were the highest. Consequently, the experiments with one neighbour were the most suitable configuration choice while the main interest was to predict and thus avoid unsafe situations. Furthermore a higher number of neighbours means a worsening of the false positives, although it seems that values over three or more neighbours do not affect the ability of predicting unsafe situations.

5 Conclusions

In this paper we have introduced the implementation of two similarity measures for event sequences within the *myCBR* software. Our objective was to provide the necessary functionalities to enable myCBR to develop CBR systems in problem domains where event sequences are of importance. Those domains often contain activities that follow sequences, such as work flows or sensor data logs. For now we have implemented two similarity functions in the *myCBR* software: the Pirjo's *Edit Distance* and the *Match & Mismatch* measures for event sequences. In order to demonstrate the usefulness of these measures, we have built a synthetic case-base that contained cases representing the "normal" and "not normal" activities of an industrial furnace. We then performed a set of experiments on this case-base which showed low error rates with both measures. Furthermore, despite the fact that this initial experimentation is based on synthetic data, the results showed that it is possible to predict unsafe situations in advance, employing either the Edit distance or the Match & Mismatch measures.

However, even though getting good results in our experiments, the remaining future work has to use data from real scenarios, such as surveillance of elderly people at home or monitoring of patients in the hospital to predict a patient's future health status. Next to these testing of our approach on real world data

it is also still necessary to compare the results from our approach with those achieved by other technologies, such as for instance, rule-based systems.

Acknowledgements. This work was partially funded by the Seneca Research Foundation of the Region of Murcia under project 15277/PI/10, and by the Spanish Ministry of Science and Innovation+European FEDER+PlanE funds under the project TIN2009-14372-C03-01.

References

1. A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues , methodological variations, and system approaches. *AI Communications*, 7:39–59, 1994.
2. OddErik Gundersen. Toward measuring the similarity of complex event sequences in real-time. In *Case-Based Reasoning Research and Development*, LNCS. 2012.
3. S. Kapetanakis, M. Petridis, B. Knight, J. Ma, and L. Bacon. A case based reasoning approach for the monitoring of business workflows. In *Case-Based Reasoning. Research and Development*, LNCS, pages 390–405. 2010.
4. Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10:707–710, 1966.
5. H. Mannila and P. Moen. Similarity between event types in sequences. In *DataWarehousing and Knowledge Discovery*. Springer Berlin Heidelberg, 1999.
6. M. Minor, A. Tartakovski, and R. Bergmann. Representation and structure-based similarity assessment for agile workflows. In *ICCBR*, pages 224–238, 2007.
7. P. Moen. *Attribute, Event Sequence, and Event Type Similarity Notions for Data Mining*. PhD thesis, University of Helsinki, 2000.
8. H. Obweger, M. Suntinger, J. Schiefer, and G. Raidl. Similarity searching in sequences of complex events. In *RCIS, 2010*, 2010.
9. M. Sanchez-Marre, U. Cortes, M. Martinez, J. Comas, and I. Rodriguez-Roda. An approach for temporal case-based reasoning: Episode-based reasoning. In *Case-Based Reasoning Research And Development*, LNCS, pages 465–476. 2005.
10. A. Stahl and T. Roth-Berghofer. Rapid prototyping of cbr applications with the open source tool mycbr. In *Advances in Case-Based Reasoning*. Springer Berlin Heidelberg, 2008.
11. K. Vrotsou. *Everyday mining: Exploring sequences in event-based data*. PhD thesis, Department of Science and Technology Linkoping University, 2010.
12. K. Vrotsou and C. Forsell. A qualitative study of similarity measures in event-based data. In *Human Interface and the Management of Information. Interacting with Information*. Springer Berlin Heidelberg, 2011.
13. I. Watson. Is cbr a technology or a methodology? In *Tasks and Methods in Applied Artificial Intelligence*. 1998.
14. M. Wolf and M. Petridis. Measuring similarity of software designs using graph matching for cbr, 2008.
15. K. Wongsuphasawat, C. Plaisant, M. Taieb-Maimon, and B. Shneiderman. Querying event sequences by exact match or similarity search: Design and empirical evaluation. *Interact. Comput.*, 24:55–68, 2012.
16. K. Wongsuphasawat and B. Shneiderman. Finding comparable temporal categorical records: A similarity measure with an interactive visualization. In *Visual Analytics Science and Technology, 2009. VAST 2009. IEEE Symposium on*, 2009.