To appear in 6th ACM Conference on Information and Knowledge Management (CIKM'97), Las Vegas, November 1997.

A Framework for the Management of Past Experiences with Time-Extended Situations

Michel Jaczynski

INRIA Sophia-Antipolis, Action AID
2004 route des lucioles - BP 93
06902 Sophia Antipolis Cedex, FRANCE
e-mail: Michel.Jaczynski@sophia.inria.fr

Abstract: In the context of knowledge management, we focus on the representation and the retrieval of past experiences called *cases* within the Case-Based Reasoning (CBR) paradigm. CBR is a problem-solving method based on the reuse of past experiences that represent pieces of knowledge. The first step of this kind of reasoning is the retrieval from the memory of relevant cases to solve a new problem by reuse and adaptation. In this paper, we propose a framework to be used in the design of CBR systems that need a retrieval of cases with time-extended situations. Time-extended situation assessment is required in many different applications such as automatic control, medical problem solving, process supervision and forecasting. Inside our framework for past experiences management, we propose a representation model that mainly handles situations described by a set of sequences of events and/or sampled data. We also propose a complex retrieval strategy that integrates the retrieval of different types of knowledge: abstract, concrete and potential cases. Our framework, implemented in an object-oriented way, can be applied to a general class of problems and mainly supports the knowledge discovery and explanation.

1. INTRODUCTION

In the context of information and knowledge management, we have studied the management of past experiences called *cases* within the underlying Case-Based Reasoning paradigm. Generally speaking, Case-Based Reasoning (CBR) is a problem-solving method based on the reuse of cases [1]. A case basically represents a problem situation, the solution that has been applied (or a way to compute it), and sometimes its evaluation. Cases must be structured and indexed into a memory in order to be reused when similar problems are encountered. The first step of the reasoning is the retrieval through indexes of relevant cases which are somehow similars, or match partially the current problem situation. The goals of others steps are the reuse of the past solution by adaptation, the evaluation of the proposed solutions and finally the learning of this new experience in the memory for future reuse. Thus, Case-Based Reasoning addresses the storage and retrieval of knowledge, and reuse of knowledge in a connected way. Case-Based Reasoning leads to a new generation of databases where data are not only stored for saving purposes but are also intended to be reused, which is a key point in many problem-solving activities.

In this paper, we will focus on the representation and the retrieval issues of cases with *time-extended situations*. We intend to use not only the current state of the observed process or activity but also its past history during the situation assessment and the retrieval step. Many approaches in CBR have taken into account only an *instantaneous situation* described by a finite set of data that represents the state of the world at a particular time, but in a lot of real world problems it is not enough. In medical problem solving, the pattern of patient state changes is usually more important than a particular state [14], and medical records have to be consulted. When you want to forecast or to understand the behaviour of an intelligent agent such as a human being, you have to analyse temporal sequences of interactions [18].

Few existing works in Case-Based Reasoning have tried to represent and use time-extended situations inside cases in different applications: robot control [17], process forecast [19,16], process supervision [8], trend prognoses for medical problems [21] and medical risk detection and forecast [5]. In these works, the representation and the retrieval of this kind of situations have shown many specificities compared to standard and instantaneous situations in CBR, mainly because:

- the process or activity is observed by one or multiple streams of data coming from different sensors or sources,
- the data may be incomplete and noisy,
- the observation data can be obtained on a regular basis (sampled data) or without any fixed frequency (event driven observation),
- the retrieval must integrate the matching of data histories.

Instead of having an approach restricted to a specific application, our goal is to cope with a class of problems and we propose a framework for the management of past experiences that addresses the representation and retrieval of cases with time-extended situations. First, we will analyse the requirements of our approach compared to related CBR works. Then, we will present our time-extended situation framework composed of a representation model and a retrieval strategy. Finally, we will illustrate the use of our framework for a plant nutrition control problem.

2. REQUIREMENTS OF OUR FRAMEWORK

Our goal is to improve the management of past experiences based on our knowledge of different applications such as plant nutrition control (cf. §4) and on the analysis of the limitations of related works in CBR [17,19,16,8,21,5]. First, we will define the requirements to cope with a general class of problems in order to improve the reusability of the proposed management techniques. Then we will address four other issues that we want to integrate in order to extend the existing representation and retrieval methods.

2.1. Design of a Reusable Framework

Related CBR works [17,19,16,8,21,5] are application-oriented, even if some of them partially present more general results [19,16,5]. In our approach, we want to design a reusable framework for the management of cases with time-extended situations. For this purpose, we have defined the features of the aimed class of problems. These features have not been addressed as a whole yet by these related works:

- the process or activity is observed through multiple data streams, with incomplete and noisy data, and where the data occurrences are accurate and ordered,
- forecast of some streams of data may be available,
- both sampled data and events must be handled,
- instantaneous data must be included in a situation to represent constant operational properties,
- the retrieval must take into account the domain knowledge and the problem context.

With these requirements, the goal of our framework is to provide a representation model, that can be applied to different applications, and a retrieval strategy. The retrieval strategy must define a structured sequence of retrieval steps in which the specific methods of matching could be defined in regard to the application domain knowledge. The application will then be able to retrieve relevant cases identified by the behaviour of a set of data streams which are similar to the current problem behaviour.

2.2. Cases for Knowledge Discovery

We want to use our framework as a knowledge discovery and explanation tool and we must be able to confront acquired knowledge with domain experts (with poor domain knowledge). For this reason, a case must be related to only one *explicit* concrete situation, it must explain and evaluate its outcome and may be linked to similar cases of success or failure. This goal is not satisfied by the works based on *non-explicit* situations [19,17] where a situation is a dynamic part of the history stored in the case because features of each situation cannot be retained. An explicit situation must be used as in [16,8,21] but must also satisfy all other requirements (cf. §2.1). In addition, the situation has to represent only the relevant data over time used for reasoning, and the selection of relevant data may become fine-grained and case-dependent due to the increasing domain knowledge.

2.3. Use of Domain Scripts

These scripts represent temporal relations between observation data and can be used to define typical scenarios that are well defined in the domain knowledge. A script is in fact a type of abstract cases. Abstract cases are also used in other works [17,21] but do not cope with both event and sampled data.

2.4. Use of Situation Restrictions

These restrictions are a kind of viewpoints and they can be identified semantically even in a poor domain knowledge. These restrictions are used mainly in the retrieval to compute the matching on a smaller part of the situation. In REBECAS [19], viewpoints are used to select only specified entities, and in [21], different trend descriptions represent abstractions of past behaviours over pre-defined sets of time periods. The first approach uses restrictions on the observation variable space and the second one on the time space, but we need to extend this notion by taking both spaces into account.

2.5. Efficient Behavioural Data Storage

When the system observes a process with sensors for example, the amount of data can become very large. In SINS [17], the storage problem is addressed by the creation of more abstract cases which summarise the knowledge coming from old cases and new experiences, but the process histories not taken into account are forgotten. This problem is not really addressed in the other approaches and all the relevant observations are stored into memory without any specific treatment. In our approach, we think that all the relevant knowledge cannot be known at a given instant as in SINS and we may need to consult not only the old cases but also the process histories, in order to update the system knowledge. Thus, we need to store many histories and the representation must provide a way to reduce the storage space needed for behavioural data.

3. PROPOSED FRAMEWORK FOR REPRESENTATION AND RETRIEVAL

From the above requirements, we first propose an object-oriented representation model based on the separation of the observation data inside the *records* from the knowledge represented in *cases*. Based on this representation, we have designed an open retrieval strategy for cases indexed by time-extended situation. This framework has been implemented in the CBR*Tools software library [12], and we will finally point out the main features of the implementation.

3.1. Observation Data Representation

Our goal is not to define a general purpose observation data representation as in temporal or time series databases approaches [22,23,6], but we need to define an abstract representation model to make explicit the assumptions needed by our framework on these data.

3.1.1. Variables and Time Series

We assume that process and its environment are observed through a set of variables and their evolutions are recorded into different *time series*. The values of variables must be preprocessed to partially cope with missing and noisy data. Signal processing methods and domain knowledge can be used to generate missing values, to delete erroneous data and to reduce noise. In our framework, no assumption is made either for the data type of each variable or for the frequency of data occurrence. A time series $\overline{X}$ represents the evolution over time of one variable X. $\overline{X}$ is an application from a finite set $T_{\overline{X}}$ to a set $E_{\overline{X}}$ defining the domain value of the time series. For $t \in T_{\overline{X}}$, $\overline{X}(t)$ denotes the value taken by the variable X at the date t. $T_{\overline{X}}$ is a subset of $\Omega_{\overline{X}}$, a totally ordered set. $\Omega_{\overline{X}}$ represents our model of the real-world time at the level of time series.

We have defined two types of time series: *sampled* and *event-based* time-series. In sampled time series, the duration between two following elements of $\Omega_{\overline{X}}$ and $T_{\overline{X}}$ are constant (cf. Figure 1).

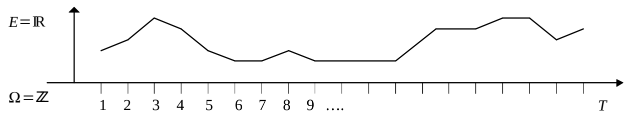

Figure 1: Example of a numeric sampled time series

When a time series is not sampled, it is an event-based time series and each value of a variable is called an *event*. For example, if the events have a meaning of a state change, we obtain a concise history (cf. Figure 2)

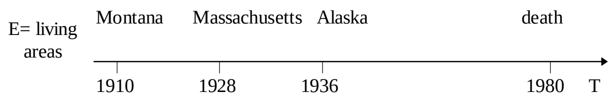

Figure 2: Concise history of living areas (adapted from [25])

In addition, independently of being sampled or event-based, a time series may have forecast observation following the past real observations. To take into account all these definitions and constraints, we propose an objet-oriented class hierarchy that abstracts the data access to the time series in the class Time Series (cf. Figure 3). This abstraction defines the interface needed by our framework while leaving open the implementation. Due to the composition of two Simple Time Series, the class Forecast Time Series allows easy updates of forecast data while uniform access methods are maintained. Other classes have been derived from Simple Time Series to implement sampled or event based time series.

Compared to time series database systems (TSMS) [22,23,6], $\Omega_{\overline{X}}$ is similar to a *calendar* [22] and represents the set of time points derived from a *time granularity* [23], $T_{\overline{X}}$ represents the set of *data points* [23], but forecast data in time series are not explicitly supported. It may be useful to integrate in our framework a TSMS for a low level management. This interface could be done easily by defining new specialised classes.

3.1.2. Records

A record R is composed of the time series of all variables between two precise dates $(t_0, t_f) \in \Omega_R^2$ and a record context (cf. Figure 3). All time series must use the same time model Ω_R. The notion of records is similar to a *group* of time series [6] and the record context is a king of *group header*, but in a record we assume more constraints on time series. Our model of records presents two main advantages:

- *the variable histories are segmented according to the record context*. The observation of a process can indeed usually be decomposed in periods of time which do not have any causality relations. For each period of time constant and specific parameters can be expressed, such as the type of the process operational constraints or the type of the environment. For instance, if we observe a doctor decision making process, we will create a record for each patient, and the name of the patient will be a part of the record context.
- *the storage of time series can be optimised*. The record manages also the storage of all time series it contains. The numerical time series can be approximated (compression with lost) through segmentation techniques [9] for example. The symbolical time series can be compressed on the finite alphabet defined by the events of a record, with Huffman coding [10] for instance. In addition, the records enable data sharing between overlapping time-extended situations. Thus, the storage of time series can be greatly reduced.

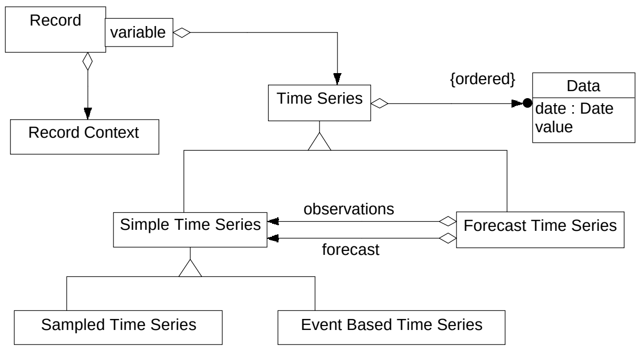

Figure 3: Classes for records and time series representations using the OMT notation [20]

3.2. Knowledge Representation

Knowledge is stored in each case which retains a piece of experience based on observation data. In our framework, each case is related to a time-extended situation that defines when the case is relevant. In order to handle different types of knowledge and knowledge discovery, we also use three types of cases: *abstract, potential* and *concrete*.

3.2.1. Time-Extended Situation Representation

A time-extended situation must define the relevant features among the available data in a record in such a way that these features can be identified at the current reference date, when a new problem has to be solved. This identification must also consider different kinds of data (numeric, symbolic), different kinds of times series (sample or event-based), and may concern data before the situation reference date or after it, as far as forecast data are taken into account in the current problem situation. The record context may also defines some relevant instantaneous features for the reasoning.

For these reasons, we propose the object oriented-model depicted in Figure 4, for the representation of time-extended situation. A time-extended situation has an optional instantaneous part and a behavioural part. The instantaneous part contains the record context. The behavioural part refers to a record at a precise *reference date*, defines for each variable a set of elementary behaviours, and may relate these behaviours through a set of temporal constraints (cf. Figure 4). Each elementary behaviour can represent a single item of data or a sequence of data of a time series.

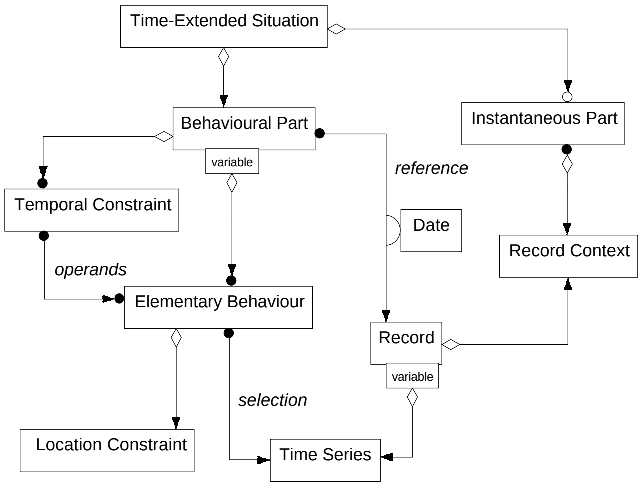

Figure 4: Time-extended situation representation model

In order to identify elementary behaviours in the new problem relatively to the reference date, we associate to each behaviour a location constraint. We propose three kinds of location methods (cf. Figure 5):

- *by position*: the number of elapsed events is specified from the reference date.
- *by elapsed time*: the duration from the reference date is specified.
- *by value*. The value could be an event, a sequence of events or a sequence of sampled data. This value is specified by a position from the reference date in the original situation. A similarity measure must be defined in order to locate the best data or the sequence of data inside a time series when applied in a new situation. An additional time range may be added to constraint the search space.

After the definition of these elementary behaviours, we obtain a uniform representation of more abstract pieces of the situation behaviour which can be connected through explicit temporal relation constraints. Each elementary behaviour has a beginning and an ending date which can be related by using a temporal logic formalism on intervals [2].

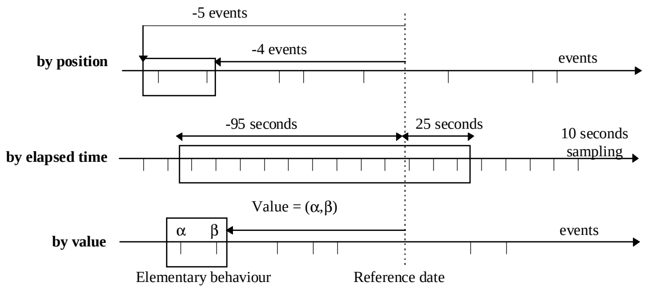

Figure 5: Elementary behaviour location constraints

The proposed object-oriented model is very flexible as we are able:

- to select a set of relevant data sequences,
- to select precise events,
- to describe explicit temporal constraints between relevant data,
- to define an optional instantaneous part of a time-extended situation.

For instance, we can represent a behavioural part which may be:

– simple : « *The situation A is defined at the instant t_0 by the five last values of all the time series* ». In this case, the elementary behaviours are defined by position and there aren't any temporal constraints.

– or complex : « *The situation B is defined at the instant t_1 by the values of the variable V_1 in the last 10 seconds, by a high increase of the value V_2, and by the event α followed by β on the variable V_3 , β occurring during the increase of V_2* ». This situation (cf. Figure 6) illustrates how it is possible to take event-based (V_1 and V_3) and sampled (V_2) time series into account in the same situation.

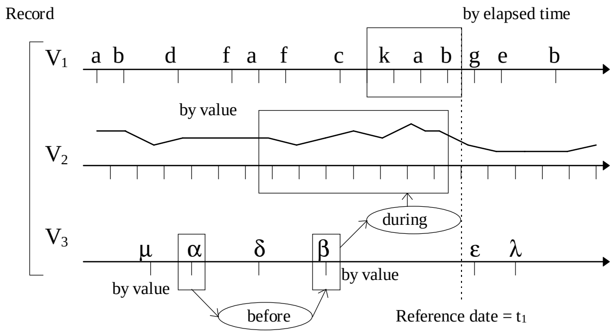

Figure 6: Typical behavioural part of a time-extended situation

3.2.2. Three Types of Knowledge: Abstract, Potential and Concrete Cases

In Case-Based Reasoning, abstract cases come from the domain knowledge or they are built by manual or automatic generalisation [3]. In the current version of our framework, we do not address generalisation, but we use abstract cases to represents domain scripts. Thus, abstract case situations are related to a record through a virtual date. This record does not come directly from real observations because the time series are built based on the domain knowledge.

We introduce the notion of *potential* cases. Potential cases do not have a concrete representation, and the knowledge they represent is hidden inside the record data that are still stored in memory. They can be identified by a direct search inside the records according to a *potential case template*. This template defines typical location and temporal constraints, and can be instanciated at a given reference date. These cases cannot be used directly, but due to some new problems, potential cases could become explicit as concrete cases (cf. §3.2).

Concrete cases are the basic type of cases which represent an explicit elementary experience and is identified by a time-extended situation based on real observation data.

3.3. Retrieval Strategy

Based on our time-extended situation representation, we propose a case retrieval strategy with an open representation and implementation.

3.3.1. Problem Representation and Retrieval Goal

In Case-Based Reasoning, the retrieval step must identify one or more cases that can be reused for the resolution of the current problem. As it is very rare to find a case that matches exactly the current problem situation, the case-based reasoner must retrieve cases that are similar to the current problem situation.

In our approach, a problem is defined by a time-extended situation that may include forecast time series in its record. The behavioural part of the problem situation does not integrate any elementary behaviours nor temporal constraints, and all the data of the record are accessible. On the other side, each case defines a situation with its own relevant structure, thus the retrieval process must map the situation of the cases to the current problem situation in order to evaluate the similarity of the cases.

3.3.2. Strategy Definition

Using our representation, we propose a retrieval strategy which is divided into three main steps and takes abstract, concrete and potential cases into account. This retrieval process is based on the following heuristics: it is better to retrieve an abstract case than a concrete case because abstract cases come from the domain knowledge; and it is better to retrieve a concrete case than a potential case because a concrete case has been reused at least once.

1. *Abstract case retrieval.* Abstract cases are more reusable and represent typical scripts. Even script with complex temporal relations can be identified efficiently [15]. If the matching of the current problem with an abstract case is not sufficient, then the retrieval process must continue.
2. *Concrete case retrieval.* This step is divided as follows:
   - **a.** *Filtering on the record context.* The record context defines properties of the process during a record. The purpose of this sub-step is to select records that have a compatible context with the current record context: it is useless to retrieve cases related to records where the operational properties are totally different.
   - **b.** *Restriction filtering.* In our model, a restriction takes both variable space and time space into account by defining a filter on variables and time horizons. Based on location constraints, a time horizon is constrained by position or by elapsed time relatively to the reference date of a situation. Restrictions are used to filter the case base on a restriction of the situation (goals, most important relevant data, time horizons). The aims of this sub-step are to ensure a minimal relevance of retrieved cases and speed up the retrieval process.
   - **c.** *Filtering based on elementary behaviours.* For each case, the elementary behaviours are identified in the current problem through the location constraints. Then the case similarity is computed, based on similarity measures for each variable: numerical data similarity, or event sequence similarity (with ordered or unordered constraints, missing or added event tolerances) based on elementary event similarities.
   - **d.** *Selection based on temporal constraints.* Temporal constraints are checked for each remaining cases and ordered according to a similarity function.
3. *Potential case retrieval.* If matching cases are still not found, a selection of potential cases is done. First, the potential case templates are selected. Then the potential cases are identified by a linear search inside the filtered records from step 2/a. Finally, concrete cases are created for the best matches.

3.3.3. Retrieval Representation

We propose an objet-oriented model based on *indexes* which are used to build our complex retrieval strategy. An index is a structure that provides access to a set of cases according to a given problem situation. Indexes may be *simple* or *composite* (cf. Figure 7). Simple indexes are used to represent an elementary access structure such as standard K-Nearest Neighbours, K-d Trees [24], fuzzy prototypes [11]. Then, these structures can be composed to build a more complex structure through composite indexes with alternatives or in a sequential manner.

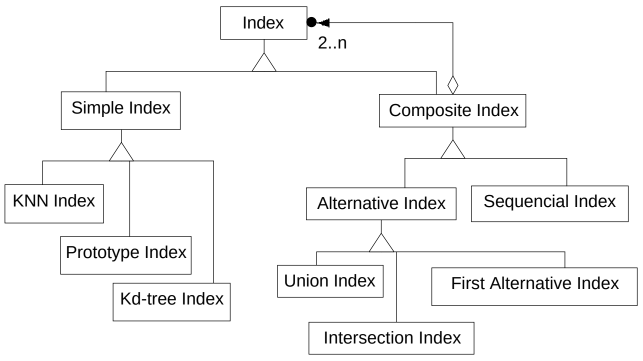

Figure 7: Class diagram for index representation

Our strategy is represented by a First Alternative Index composed of three Sequential Index. The three main steps of the retrieval are indeed alternatives and the retrieval result is returned by the first alternative which finds at least one matching case. Each alternative can be implemented inside a sequence by specific indexes depending on the application. Thus our framework is open and may be extended by the definition of new indexes.

3.4. Framework Implementation

We have implemented our framework in a software library, called CBR∗Tools, using Java programming language. The design of this library has been done using the Object Modelling Technique [20]. Object-oriented design is central in our framework, since we provide a set of classes that can be used by composition and/or by inheritance in order to apply the framework to a specific application. The CBR∗Tools library provides:

- classes to represent records, sampled or event time series, elementary behaviours and temporal constraints;
- classes that manage the retrieval strategy in an open way since we can define and add new steps;
- classes that implement indexing and retrieval techniques used to build the retrieval process: crisp and fuzzy prototypical filtering [11], K-d trees [24], and the standard K-Nearest Neighbour (KNN) method with an extendible class hierarchy of similarity functions, as well as potential case retrieval that we have defined in our work;
- management of cases, records, indexes, and potential case templates in collection classes, that use the standard serialization package of Java for object persistence.

4. FRAMEWORK ILLUSTRATION ON A PLANT NUTRITION CONTROL PROBLEM

In this section, we present an example from the application of our framework to a plant nutrition control problem. First we will introduce the problem features, then we will describe the instanciation of our framework for the representation of observation data and knowledge, and for the retrieval process.

4.1. Plant Nutrition Control Problem

The aim of fertigation techniques are to supply plants with nutrients and irrigation water so that the plants receive exactly what they need while the production constraints are satisfied. In the approach taken at INRA[1] [4], the purpose is to define an automatic feedback controller that computes every day the concentration of a fixed nutrient mixture in the irrigation water in order to maintain the concentration of the leachate around a defined set point. The leachate is the water collected after the plants have absorbed what they needed. A controller based on a simple formula has been used [4] but the results can be improved.

In this problem the domain knowledge is very poor and can only be represented by a set of qualitative relations. Above all, the experts have observed fuzzy reaction delays: a change of the nutrient mixture concentration has an impact on the leachate concentration from 2 days in summer to 5 days in winter. The problem situation cannot be reduced to a fixed set of values, but must take into account the past histories of the observation variables, and the forecast values of the sunlight radiation for the future days. As it is pointed out in [4] the weather forecast is required for improving the plant nutrition control procedure. We have applied our framework to this problem to build a knowledge discovery workshop where we can:

- automatically associate regulation actions to a time-extended situation inside a case,
- update manually if necessary the knowledge of a case by modifying the time-extended situation description,
- and aid the operator to control the plant nutrition by adapting the past control sequences.

4.2. Representation of Observation Data and Knowledge

As we have seen previously, our framework supports the representation of observation data, structured in variables, time series and records, and the representation of the knowledge in different types of cases.

4.2.1. Variables, Time Series and Records

In this application, the process and its environment are observed through different sensors: temperature, humidity, volume and concentration of the leachate and the nutrient solution. The concentrations are evaluated by the electrical conductivity of the solution. As it is done in [4], the different values of each sensor are aggregated to represent the relevant behaviour over a day. This leads us to define a set of relevant variables. The controlled variable is the nutrient solution electrical conductivity for a day (NEC), and the feedback variable (FD) is the difference between the leachate electrical conductivity and the set point. Disturbances to the environment are observed using other three variables: the maximum day temperature (DTM), the sunlight radiation sum (SRS), the maximum humidity (HM). A record is composed of the time series of the five variables. Each time series is composed of numeric values for each day between the beginning and the ending date of the record.

4.2.2. Cases

A concrete case contains: a concrete situation, the nutrient electrical conductivity used to control the process, and the evaluation. The case evaluation is computed on the basis of the feedback variable evolution. As we have seen previously, there is a control delay correlated to the sunlight radiation from 2 to 5 days. This means that a case situation will not be defined by the same relevant data whether the sunlight radiation is high or low. Our representation model provides a simple way to select elementary behaviours inside the record for each case. Thus, four potential case templates are used in order to select the past values from all variables from 2 days in winter up to 5 days in summer. In templates, elementary behaviours are identified by position from the reference date. During the reasoning concrete cases will be created on the basis of retrieved potential cases. Concrete cases may be then updated by the user and new

[1] French National Research Institute in Agriculture.

elementary behaviours and temporal constraints can be added (cf. Figure 8).

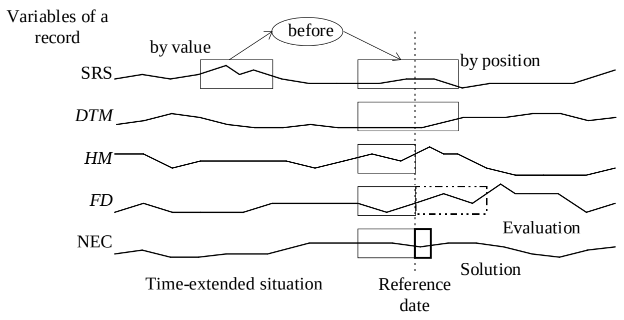

Figure 8: Example of a plant nutrition control case

Abstract cases are used to describe emergency situations: when the feedback variable goes outside upper and lower bounds, the case-based reasoner must inform the operator.

4.3. Retrieval of Plant Nutrition Control Cases

We have applied our retrieval strategy and we have selected the appropriate retrieval technique for each step:

1. *Abstract case retrieval.* A set of abstract cases where the system is not assumed to be under control has been defined. The retrieval is made by a KNN search with a threshold similarity. If there were no abstract cases with enough good similarity, the retrieval must continue.
2. *Concrete case retrieval.*
   a. *Filtering on record context.* The record context is defined by the leachate electrical conductivity set point which is constant for one record. The record context filtering step is made by an exact search query on the current electrical conductivity set point because we have not enough knowledge yet to reuse cases of different set points.
   b. *Restriction filtering.* We have defined the *short term* restriction that reduces the past histories to two days for a subset of variables. This restriction is used to select past histories that are relevant in all cases: in winter and in summer. As the number of values identified by the restriction is fixed, K-d tree similarity-based retrieval is used [24]. Cases that are under the filtering similarity threshold are no more taken into account.
   c. *Filtering based on elementary behaviours.* A fine grained filtering base on the remaining cases is done using KNN with a similarity based on a mean squared difference between vectors (similar to [17]).
   d. *Selection based on temporal constraints.* The selection based on temporal constraints uses a KNN index based on a similarity that computes the ratio of the number of satisfied constraints with the total number of constraints.
3. *Potential case retrieval.* Finally, if no case were found, and especially at the beginning of the system life, the potential case selection is made using the potential cases templates. The selection of one template depends on the current sunlight radiation. Then the templates are used to scan the records identified in the step 2/a, in order to identify relevant potential cases and discover new concrete cases.

5. CONCLUSION

In the CBR paradigm, the management of knowledge, represented in cases, is a key part of the reasoning. We have proposed a framework for past experience management that takes into account the representation and the retrieval of cases with time-extended situations. This kind of knowledge management is often required in applications which deal with an evolving process or activity. The originality of our work resides in the design of a framework that integrates all the following features according to our requirements:

- We propose a framework that addresses a general class of problems, and mainly it handles forecast data in the problem description, and supports sampled and event-based behavioural data through the definition of two types of constraints: location constraints and temporal relations. In addition, the framework has been implemented in an object-oriented way to enable its specialisation to a wide range of applications.
- The framework supports knowledge discovery and explanation through fine-grained situation description and the retrieval of potential cases when needed.
- We have integrated domain scripts as abstract cases in the first step of our retrieval strategy.
- We have used filters on both variable and time spaces to ensure a minimal relevance of retrieved cases and to speed up the retrieval.
- We have proposed the separation of concrete cases and observation data in records in order to reduce the storage needs while keeping experience knowledge through potential cases.

In order to extend the supported class of problems and to improve the framework, we plan to integrate: the representation of temporal granularities [14], learning methods to update the situation of cases with the increasing knowledge, and other index algorithms [7]. We are also studying the validation of our framework to other applications inside our class of problems: collaborative argumentation support on Internet during a decision making activity [13], and assistance to a user during an Internet browsing session. In the latter application, cases identified by the past behaviours of users must be represented, indexed and retrieved in order to determine the assistance actions.

ACKNOWLEDGEMENT

We would like to thank Brigitte Trousse for her useful comments and suggestions.

REFERENCES

[1] A. Aamodt and E. Plaza. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System. *AI Communications*, 7(1) :36–59, March 1994.

[2] J. F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154, 1984.

[3] R. Bergmann and W. Wilke. On the role of abstraction in case-based reasoning. In I. Smith and B. Faltings, editors, *Advances in Case-Based Reasoning*, volume 1168 of *Lecture Notes in Artificial Intelligence*, pages 28–43. Springer, 1996.

[4] R. Brun, B. Paris, and I. Hammelin. Fertigation management of rose plants grown in greenhouse on rockwool. *Adv. Hort. Sci.*, 7:109–111, 1993.

[5] M. Bull, G. Kundt, and L. Gierl. Case-based risk detection and forecasting in a geographic-medical system. In R. Bergmann and M. Wilke, editors, *German Workshop on Case-Based Reasoning*, pages 59–64, March 1997.

[6] W. Dreyer, A. K. Dittrich and D. Schimdt. Research perspectives for time series management systems. *SIGMOD Record*, 23(1) :10–15, March 1994.

[7] C. Faloutsos, M. Ranganathan and Y. Manolopoulos. Fast subsequence matching in time-series database. *SIGMOD Record*, 23(1) :419–429, May 1994.

[8] B. Fuch, A. Mille, and B. Chiron. Operator decision aiding by adaptation of supervision strategies. In M.Veloso, K.D. Althoff, and M. M. Richter, editors, *Case-Based Reasoning Research and Development*, Lecture Notes in Artificial Intelligence, pages 23–32. Springer, 1995.

[9] P. Garnesson and G. Giraudon. Polygonal approximation: overview and perspective. Technical Report n°1621, INRIA, June 1991.

[10] G. Held, and T.R. Marshall. *Data compression*. J. Wiley and Sons, 1991.

[11] M. Jaczynski and B. Trousse. Fuzzy logic for the retrieval step of a case-based reasoner. In M. Keane, J.P. Haton, M. Manago, editors, *EWCBR-94 : Second European Workshop on Case-Based Reasoning*, pages 313-321, Chantilly, France, novembre 1994.

[12] M. Jaczynski and B. Trousse CBR*Tools: an object oriented library for indexing cases with behavioural situation. Research Report n°3215, INRIA, July 1997. in French.

[13] N.I. Karacapilidis, B. Trousse and D. Papadias. Using Case-Based Reasoning for argumentation with multiple viewpoints. In D.B. Leake and E.Plaza, editors, *Case-Based Reasoning Research and Development (ICCBR'97)*, volume 1266 of *Lecture Notes in AI*, pages 541–552. Springer, 1997.

[14] E. T. Keravnou. Modelling medical concepts as time-objects. In M. Stefanelli and Jeremy Wyatt, editors, *Artificial Intelligence in Medicine*, volume 934 of *Lecture Notes in Artificial Intelligence*, pages 67–90. Springer, 1995.

[15] M. Ghallab and A. Mounir-Alaoui. Managing efficiently temporal relations through indexed spanning tree. In *Proc. 11th IJCAI*, pages 1297–1303, 1989.

[16] G. Nakhaeizadeh. Learning prediction from time series: a theoretical and empirical comparison of CBR with some other approaches. In *Topics in Case-Based Reasoning*, volume 837 of *Lecture Notes in Artificial Intelligence*, pages 65–76. Springer, 1994.

[17] A. Ram and J.C. Santamaria. Continuous case-based reasoning. In *AAAI Case-Based Reasoning Worksop*, pages 86–93, 1993.

[18] F. E. Ritter and J. H. Larkin. Developing process models as summaries of HCI action sequences. *Human - Computer Interaction*, 9:345–383, 1994.

[19] S. Rougegrez. Similarity evaluation between observed behaviours for the prediction of processes. In *Topics in Case-Based Reasoning*, volume 837 of *Lecture Notes in Artificial Intelligence*, pages 155–166. Springer, 1994.

[20] J. Rumbaugh, M. Nlaha, F. Eddy, W. Premerlani, and W. Lorensen. *Object Oriented Modelling and Design*. Prentice Hall, 1991.

[21] R. Schmidt, B. Heindl, B. Pollwein, and L. Gierl. Abstraction of data and time for multiparametric time course prognoses. In I. Smith and B. Faltings, editors, *Advances in Case-Based Reasoning*, volume 1168 of *Lecture Notes in Artificial Intelligence*, pages 377–391. Springer-Verlag, 1996.

[22] A. Segev and R. Chandra. A data model for time-series analysis. In *Advanced Database Systems,* volume 759 of *Lecture Notes in Computer Sciences,* chapter 10, pages 191–212. Springer, 1993.

[23] A. Segev and A. Shoshani. A temporal data model based on time sequences. In A.U. Tansel *et al.*, editors, *Temporal Databases: Theory, Design and Implementation*, Benjamin/Cummings, chapter 11, pages 248–270, 1993.

[24] S. Wess, K.D. Althoff, et G. Derwand. Using K-d Trees to improve the retrieval step in case-based reasoning. In S. Wess, K.D. Althoff and M. M. Richter, editors, *Lecture Notes in Artificial Intelligence, Topics in Case-Based Reasoning*, pages 167-181, Springer, 1994.

[25] B. C. Williams. Doing time: putting qualitative reasoning on firmer ground. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 105–112, 1986.