

# An HTTP based server for Temporal Abstractions

C. Larizza, R. Bellazzi, G. Lanzola  
Dipartimento di Informatica e Sistemistica, Universita' di Pavia,  
via Ferrata 1, 27100 Pavia, Italy  
e-mail: cri, ric, giordano@aim.unipv.it

## Abstract

This paper describes an HTTP based server able to extract from multi-dimensional time series different types of patterns. It exploits a method coming from the AI field, called Temporal Abstractions, which is used to obtain a high level description of temporal data. The server has been successfully integrated within a telemedicine system for the home monitoring management of diabetic patients.

## 1. Introduction

Many monitoring problems and, in particular long-term monitoring of chronic patients, involve a great effort of the physicians to carefully analyze and interpret the large amount of clinical data collected over time. In fact, patients follow-up data are heterogeneous and present some common features that prevent from adopting classical time series analysis techniques. The most important characteristics of monitoring data are the following:

- the information is usually both of *qualitative* and *quantitative* nature;
- the data are in general drawn on an *irregular time grid* and sometimes data can be *missing*;
- the quantitative data are prone to *measurement errors*;
- the data interpretation is usually highly dependent from some contextual information, like, for example, the patient life style.

In such conditions, although the data set can be large, it is sometimes not possible to apply standard time series analysis techniques. In order to exploit the overall information available, it is necessary to resort to innovative methods, able to

combine heuristics with statistical-based data processing. In this context, an interesting approach to the analysis and interpretation of this kind of data is given by an AI based technique, called *Temporal Abstractions (TAs)*, which provides a useful and flexible method for obtaining an abstract description of multi-dimensional time series [Shahar, 1996, Shahar, 1997, Shahar 1999, Horn 1997, Larizza, 1997]. The usefulness of these methods has been proved in many medical contexts, from the long-term monitoring of infectious complications in heart transplanted patients [Larizza, 1992], to the analysis of ICU data [Horn, 1997], and finally to the management of home monitoring of diabetic patients [Shahar, 1996, Bellazzi, 1998].

In these applications these methods have been exploited both to pre-process the huge amount of raw data [Larizza, 1997] and to post-process results of a preliminary data analysis [Bellazzi, 1999]. Considering the potential usefulness of these methods in many monitoring applications, we developed a general purpose software tool that can be integrated within a distributed architecture to provide different kind of Temporal Abstractions. In particular, we have implemented a HTTP TA-server that is able to provide TAs within Internet-based distributed architectures.

In this paper we describe the different TA tasks performed by the server, then we present the communication language adopted; finally we show how the server has been integrated within a distributed system for the monitoring management of chronic patients. In the rest of the paper we'll use the term TA to indicate both the tasks performed by the TA-server and the mechanisms implementing such tasks.

## **2. Temporal Abstractions**

As previously mentioned, TAs are techniques used to extract from temporal data *patterns* meaningful for a correct interpretation of the dynamics of the system under observation. To introduce the different TA mechanisms exploited in our approach it is useful to describe the TA ontology underlying this method. The principle of TAs is to move from a *time-point* to an overall *interval* based representation of the monitoring data through mechanisms that aggregate adjacent data into abstracted episodes. Therefore, the temporal model at the base of the TA-server comprises both the temporal primitives *time-point* and *interval*.

A *time-point* is represented with its *absolute* or *relative* temporal coordinate expressed according to a *granularity* required by the specific monitoring context. An *interval* is defined by a couple of time-points representing its *end-points*. We assume also that all the data coming from the monitoring process are time-stamped events, while TAs, which are intervals, represent situations that occur and evolve during specific time periods. To cope with the requirements mentioned in the introduction of analyzing heterogeneous (numeric or symbolic) time-series and detecting specific patterns in uni- and multi-dimensional data, we defined two TA categories: *Basic* and *Complex TAs*.

*Basic TAs* allow extracting simple patterns from numeric and symbolic uni-dimensional time series. In particular, we defined *Trend* abstractions to find out *increase*, *decrease* or *stationary* episodes in a numerical time series, and *State* abstractions to extract qualitative patterns (*low*, *high*, *normal* values) from a numerical or symbolic time series. When a *State* abstraction is performed on a numerical time series, a preliminary *qualitative* abstraction of the variable considered is carried on.

*Complex TAs* allow detecting patterns of complex shapes in uni-dimensional or multi-dimensional time series. *Complex TAs* represent higher-level abstractions not directly derived from the data, but from other TAs. The inputs of a Complex TA are two series of intervals and the result is given by a set of intervals that verify a certain temporal relationship between the inputs. The relationships investigated by *Complex TAs* include all the temporal operators defined in the Allen algebra [Allen, 1984] (before, after, meets, overlaps, starts, finishes, equals, during).

Each TA is defined by a set of parameters that specify completely the pattern to be detected. Some attributes are common to all the TAs, while others are TA specific. Example of common properties are the *granularity* that specifies the maximum gap that allows two measurements to be aggregated in the same episode and the *minimum extent* that specifies the minimal time-span of an episode to be considered relevant in that context. In particular, appropriate *granularity* values allow managing also data drawn on irregular time grids. Amongst the TA specific attributes, we can mention a property of *Trend* abstractions called *minimum rate* that defines the minimum increase or decrease rate for the pattern to be detected, and a

property of *Complex* abstractions called *temporal operator* which specifies the temporal relationship to be investigated (e.g. before, after, meets).

### 3. The TA-server

We decided to implement a TA-server for enabling the access to TA mechanisms on the Internet/Intranet/Extranet by any client compliant with some basic server specifications. This allows to include in line of principle such AI-based tool in any distributed application which needs to exploit such kind of methodology. The TA-server was implemented on UNIX platform using ANSI C language, and relies on TCP and on a slightly extended version of HTTP.

#### 3.1 The communication language

The communication protocol adopted for the server is TCP which provides a two-way connection-based transmission of data. The server communicates with each client through a *socket* connection, which is established following a client connection *request* and which is accepted by the server if the client has been authorized to access the service. The client is identified by *name* and *password*. The communication language is an extension to the HTTP protocol called *Server-To-Server Protocol (STSP)* that allows a server to provide services to other servers, or to use the services provided by them [Riva, 1998]. Thanks to this mechanism, servers can become agents in a distributed computing environment. Each service *request message* contains the *message header*, indicating the purpose of the request, and the *message body*. After the server has recognized the client, it checks for the correctness of the request message syntax; then, it responds with the *response message* whose first line contains the status code telling whether the request was accepted or not.

In the following we will describe in detail the format of both the *request message* and the *response message*.

**Message Header.** It begins with the following request line:

```
STSP <service type> HTTP/1.0
```

where <service type> is an integer code which specifies the type of service requested to the server. Table 1 reports the codes of all the services available.

Code	Type of service
0	State of the server
1	Trend abstraction
2	State abstraction
3	Stationary abstraction
4	Complex abstraction

The server accepts the following headers:

**Reference-date:** represents the reference absolute time used by the server in order to manage correctly the time stamps associated to the data to be analyzed.

**Time-unit:** refers to the granularity to be used in the analysis. If the time stamps of the data are expressed as absolute time the granularity can be: D (days), H (hours), M (minutes) while N is used if the time stamps are expressed as relative times with respect the reference date .

**Agent:** name of the user-agent requesting the service.

**Password:** password of the user-agent requesting the service.

In case of service code 0, no headers are required.

**Message Body.** This part of the message can be empty or it can contain a set of values specifying the kind of pattern to detect (for example the minimum extent and the minimum slope of a decreasing pattern) and , finally, the series of time stamped data to analyze.

**Response Message.** It contains a *status line* which includes the message's protocol version and a success or error code (*status code*). If no error occurs, it contains also the number of patterns detected followed by their end-points represented with the time unit specified in the client request.

An example of request is reported below:

```
STSP 1 HTTP/1.0
Reference-date: 01-01-1995 10:00:00
Time-unit: D
Agent: Larizza
Password: Bellazzi

-1
0.5
2
4
2
9
07/11/1997 07:15:00 245
08/11/1997 07:49:00 322
09/11/1997 07:25:00 316
10/11/1997 06:58:00 201
11/11/1997 07:04:00 158
12/11/1997 06:55:00 245
13/11/1997 07:00:00 148
14/11/1997 07:02:00 87
15/11/1997 07:04:00 169
```

The first line contains the TA code for requesting a Trend Abstraction (see Table 1), while the headers indicate the Time Unit (days represented in absolute time), the Reference Date and the personal information of the user-agent.

The *message body* contains the parameters that characterize the service requested and the data to analyze. In this example the client requests a decreasing pattern (*slope* = -1), with a *minimum rate* of 0.5 units/day, of a *minimum extent* of 2 days and a *granularity* of 4 days. Since the TA mechanism which detects trends looks for the requested *minimum rate* both on the overall temporal window length and on its sub-intervals, the message body contains another parameter (in the example its value is 2) which represents the width of such sub-intervals. This parameter has been defined to be able to detect both strictly monotonic patterns and patterns containing oscillations.

The remaining lines of the request contain respectively the time series length (in the example its value is 9) and the time stamped data.

The *message response* provided by the TA-server to the request previously reported is the following:

```
HTTP/1.0 200
Number-of-episodes: 2
08/11/1997 00:00:00
11/11/1997 00:00:00
```

12/11/1997 00:00:00  
14/11/1997 00:00:00

The *status line* contains the status code 200 which indicates that the request sent by the client was correct. In this case the server provides as output the number and the list of intervals. Since the time unit specified in the request is *day*, the end-points of the episodes detected are taken at the start of the day (hour: 00:00:00).

#### **4. Application and on-field verification**

The TA-server described in this paper has been exploited in the field of an EU funded telemedicine project, called T-IDDM (Telematic Management of Insulin Dependent Diabetes Mellitus), devoted to provide patients and physicians with an Information Technology infrastructure for a better management of IDDM. The goal of the project is the development of a set of distributed web services for the overall process of IDDM management. The T-IDDM architecture is based on the long-distance communication between two modules, a Patient Unit (PU) and a Medical Unit (MU). While the PU is devoted to assisting patients in the data collection and disease self management tasks, the MU is a fully integrated service for helping the physician in assessing and revising the therapeutic protocol of each patient under control. The MU has been designed to allow the physician to store, retrieve and analyze all the patient's data that are useful for making proper therapeutic decisions. Moreover, the automated reasoning capabilities of the MU are driven by TA-based summaries [Bellazzi, 1997]. Therefore, the MU relies on specialized servers dedicated to the analysis and interpretation of home monitoring data: in particular, since TAs represent the fundamental instrument of the intelligent data analysis part of the system, the MU fully exploits the capability of the TA-server we implemented.

The MU (see [Riva, 1998] for a technical description) access the TA-server to perform the following kind of data analysis:

- *modal days extraction*, which is the extraction of the characteristic daily patterns that summarize the typical patient's behavior during a specific monitoring period. This operation is performed by counting the number of occurrences of the State

episodes for the different variables (Blood Glucose Levels, Glycosuria, Insulin), weighted by their time span.

- *extraction of State abstractions* on all the variables measured during the monitoring;
- *extraction of Complex abstractions* (e.g suspected Somogy effect, a concurrence of High pre-breakfast Blood Glucose and an Absence of Glycosuria [Larizza, 1997]),
- *post-processing of the time series* coming from structural analysis [Bellazzi, 1999].

The TA-server has been technically verified during the verification phase of the T-IDDM project [TIDDM, 1999] and is currently being used in the project demonstration phase, that involves the monitoring of 20 patients in four different medical centers in Europe (located in Pavia, Padova, Barcelona and Helsinki).

A “secure version” of the server has been implemented and verified too, and its characteristics have been described in [TIDDM, 1999]. We plan to use our server in other applications for data analysis of chronic patients, such in hearth failure and in peritoneal dialysis.

## **5. Conclusions**

TAs may represent a valuable instrument for data analysis and summarization in variety of bio-medical applications. One reason of their limited use is the unavailability of general purpose tools for the implementation of the required mechanisms. To this end we have developed a TA-server that relies on TCP and HTTP 1.0. Such server is portable on any UNIX platform and can be easily integrated in any distributed architecture. In the near future we plan to add an external shell to this server, in order to provide the user also with facilities for interfacing with standard SQL databases.

## **Acknowledgements**

We sincerely thank Alberto Riva and Carlo Mossa for their technical support. This work is part of the EU-Funded project T-IDDM (HC 1047).

## References

- [Allen, 1984] Allen J. F., Towards a general theory of action and time. *Artificial Intelligence* 23 (1984) 123-154.
- [Bellazzi, 1997] Bellazzi, R., Larizza, C., and Riva, A, Cooperative Intelligent Data Analysis: an application to diabetic patients management, in: *Intelligent Data Analysis in Medicine and Pharmacology*, 1997, N. Lavrac, E. Keravnou, Blaz Zupan eds., Kluwer, pp. 81-98.
- [Bellazzi, 1998] Bellazzi, R., Larizza, C., Riva, A., Temporal Abstractions for Interpreting Diabetic patients monitoring data, *Intelligent Data Analysis*, 2 (1998) 97-122.
- [Bellazzi, 1999] Bellazzi, R., Larizza, C., Magni, P., Montani S. and De Nicolao, G., Intelligent Analysis of Clinical Time Series by combining Structural Filtering and Temporal Abstractions, *Lecture Notes In Artificial Intelligence* 1620, W. Horn, Y. Shahar, G. Lindberg, S. Andreassen, J. Wyatt eds, pp. 261-270, 1999.
- [Horn, 1997] Horn, W., Miksch, S., Egghart, G., Popow, C., Paky, F., Effective data validation of high frequency data: time-point-, time-interval- and trend-based- methods. *Computers in Biology and Medicine*, 25 (1997) 389-409.
- [Larizza, 1992] Larizza, C., Moglia, A., and Stefanelli, M., M-HTP: A system for monitoring heart transplant patients, *Artificial Intelligence in Medicine*. 4, 111-126, 1992.
- [Larizza, 1997] Larizza, C., Bellazzi, R., Riva, A. Temporal abstractions for diabetic patients management, in: *Lecture Notes in Artificial Intelligence*, 1997, E. Keravnou, C. Garbay, R. Baud and J. Wyatt eds., Springer Verlag, Berlin, 319-330.
- [Riva, 1998] A. Riva, R. Bellazzi, M. Stefanelli and G. Lanzola, A methodology for the development of knowledge-based medical applications on the World-Wide Web, *Artificial Intelligence in Medicine* 14 (1998) 279-293.
- [Shahar, 1996] Shahar, Y. and Musen, M.A., Knowledge-Based Temporal Abstraction in Clinical Domains, *Artificial Intelligence in Medicine*. 8, 267-298, 1996.
- [Shahar, 1997] Shahar, Y., A Framework for Knowledge-Based Temporal Abstraction, *Artificial Intelligence*. 90(1-2), 79-133, 1997.
- [Shahar, 1999] Shahar, Y., Timing is everything: Temporal Reasoning and Temporal Data Maintenance in Medicine, *Lecture Notes In Artificial Intelligence* 1620, W. Horn, Y. Shahar, G. Lindberg, S. Andreassen, J. Wyatt eds, pp. 30-46, 1999.
- [TIDDM, 1999] TIDDM consortium, Deliverable 5.2, Verification phase final report, <http://aim.unipv.it/projects/tidm>.