Tempo: a general framework for temporal data processing and abstractions

Paolo Ciccarese, Cristiana Larizza
Laboratory for Medical Informatics, University of Pavia, Italy
paolo.ciccarese@gmail.com, cristiana.larizza@unipv.it

Abstract

In this paper we present Tempo, a framework for the definition, generation and execution of data processing procedures including filtering, qualitative and temporal abstractions. The overall architecture is built upon a specific data model and organized on pipelines of modules, assembled according to a data processing meta-model. Each pipeline module conforms to well defined communication rules and wraps one or more data processing algorithms. The set of algorithms provided by default as reusable blocks can be extended with custom solutions through a plug-in mechanism. The data processing procedures can be delivered both as web-services and as software library. Tempo is conceived as a general purpose framework, although it has been initially tested in the medical field. It has been integrated within a medical guidelines implementation software tool and in a general purpose web application prototype as embedded module for the extraction of temporal patterns from generic time series.

1. Introduction

Many application domains require the collection and processing of huge quantities of temporal data for different purposes. Examples include financial analysis, scientific applications (such as weather monitoring), medical applications, like long term patients clinical monitoring and on-line biomedical signals acquisition systems, and so on. However, it is difficult to identify universal procedures for analysing temporal data, since they can present very different characteristics. In general, temporal data are obtained as series of values drawn on irregular time grids, sometimes collected at times available at different granularities and affected by noise. Such great variability requires the adoption of techniques customized on the basis of the context and on the goals of the analysis. Frequently, in order to make the inspection and the processing of temporal information easy and efficient, it is useful to transform the raw data into series of patterns that summarize their temporal evolution. This kind of analysis is usually preliminary or complementary to the execution of different tasks like: visual data exploration, temporal reasoning, temporal data mining and temporal knowledge extraction. Independently from the final goal of the patterns extraction, a critical aspect of the analysis is the robustness of the results obtained, which are related to the kind of temporal data considered. The impossibility of defining a standard procedure for temporal data processing suggested the definition of a meta-model that can be configured and customized on the specific context.

In this paper, we present Tempo, a framework for the definition, generation and execution of temporal data processing procedures. It is based on a pipeline data analysis meta-model in which different kind of reusable blocks (or modules) can be assembled according to the analysis context. Each block wraps data filtering or other data processing algorithms. Communication rules and a well defined data-type system guarantee interoperability within the pipeline and provide clear contracts for enriching the provided library of blocks through the development of custom modules. Tempo already includes a set of reusable blocks for data filtering and temporal patterns extraction based on the AI technique named Temporal Abstractions (TAs) [1, 2].

The paper is organized as follows. Section 2. presents the architecture of Tempo and the analysis meta-model implemented. Section 3. introduces the algorithms available in the default library and, in particular, the Temporal Abstraction technique. Section 4. gives a short description of the data types system adopted in the framework. Section 5. explains how a Tempo component can be delivered to be integrated into an application. Finally, Section 6. reports a short discussion about similar tools and future developments.

2. Meta-model and architecture

The Tempo architecture is build around the pipeline concept. A pipeline (fig. 1) is a set of data processing elements, that we call blocks or modules, connected in series, so that the output of one element is the input of the next one.

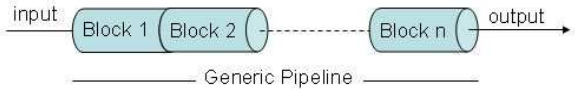

Figure 1. The generic pipeline model.

Each block represents an operator which defines the relation between an element x of a set X (the domain) and a unique element y of a set Y (the codomain) through a parameterization p of the embedded algorithm (fig. 2). Parameterization allows to decouple the pipeline from the context. In our case, X and Y can always be given the mathematical structure of metric spaces. In our model we distinguish two kind of blocks: *filters* which give an output defined within the same metric space of the input and *transformers* in which the output metric space is different from the input one.

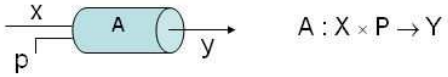

Figure 2. The generic block which implements an operator A from a domain X to a codomain Y through a parameterization in *P*.

A pipeline is build by assembling different blocks according to the data processing meta-model we defined to manage different kinds of analysis in a customizable and flexible way (fig. 3). The Tempo meta-model is organized as a sequence of zero or more filters, followed by one or more transformers followed by zero or more filters. An example of filter is given by a block embedding a noise reduction algorithm. A transformer can be, for example, a mechanism for qualitative abstraction defined as quantitative data mapping into symbolic values.

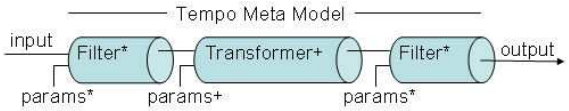

Figure 3. The Tempo pipeline meta model where '*' means zero or more and '+' one or more.

The output of Tempo is a software component for temporal data processing which is in general a manager of one or more pipelines. Different possibilities of components deployment and integration within applications will be described in the sec. 5.

3. The default modules and algorithms

Tempo includes a library of reusable blocks that can be assembled to define processing pipelines customized on the application context. They provide some standard filtering algorithms and a set of mechanisms for the temporal patterns detection that in our processing meta-model are classified as transformers.

The process of extracting specific patterns from temporal data is performed through TAs. The basic approach of this technique is to move from a time-point to an interval-based representation of longitudinal data. In our data model, the input to a TA is a collection of time-stamped entities, called events, while the output is a set of intervals, called episodes. Each interval corresponds to a specific pattern detected in the time course of the data.

We distinguish two categories of TA mechanisms:

- Basic TAs that abstract time-stamped data into intervals (input data are events and outputs are episodes);
- Complex TAs that detect temporal relationships among intervals (input and output are collections of episodes).

Basic TAs aggregate events (time-stamped data) into episodes (intervals) by detecting sequences of adjacent observations falling within a specific set of qualitative levels or showing definite patterns. We distinguish two different types of Basic TAs: State and Trend. In particular, State TAs detect episodes associated to qualitative levels of time-varying variables, like "hypoglycemia" or "severe hypertension"; while Trend TAs detect patterns, like increasing, decreasing and stationary, in a numerical time series. Every TA mechanism requires a set of parameters necessary for a complete specification of the patterns to detect. Examples are the parameter named 'gap' which represents the maximum temporal distance allowed to aggregate two measurements within the same Basic episode and the 'min-slope' and 'max-slope' parameters that define the minimum and maximum increase/decrease variable rate that trigger the detection of a Trend pattern.

Complex TAs are used to analyze two intervals series, the input of the abstraction, to detect as output the tuples of intervals that are related on the basis of a specific temporal relationship. The temporal relationships investigated correspond to the 13 temporal operators defined in the Allen algebra [3]: BEFORE, FINISHES, OVERLAPS, MEETS,

STARTS, DURING, their corresponding inverse relations and the EQUALS operator. Each operator requires a specific parameterization. For example, the FINISHES operator requires to set up the maximum distance between the two intervals starting points; while the BEFORE operator requires the maximum time gap between the related intervals. Figure 4 reports an example of both operators usage. On the top two series of input intervals are depicted while at bottom the two reported Complex TAs output tuples are shown. In the case of the BEFORE operator, three different outputs, generated according to different values of the parameter, are reported.

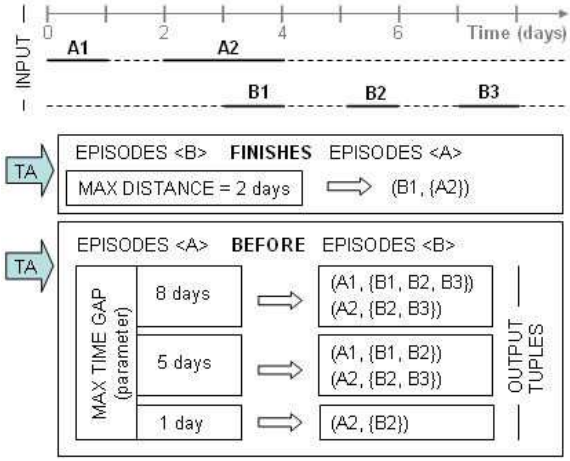

Figure 4. Two examples of Complex abstractions.

4. Data model

A fundamental step leading to the development of an effective data processing meta-model is the definition, or, in our case, the adoption, of a data types set. Our choice has been the platform Tipo developed independently as a rework of the data types framework proposed by the OpenEHR [4], HL7 [5] and JScience [6] projects. The first two projects are oriented to medical applications, while the third is more general and directed to create a synergy between different sciences.

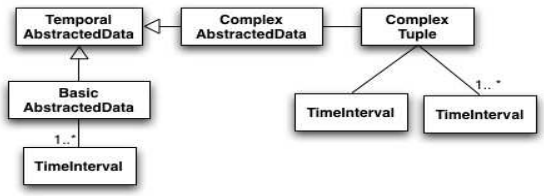

Figure 5. A portion related to Temporal Abstractions of the Tipo data model.

The data model obtained embodies not only the data types usually necessary for managing any kind of quantitative, time related or symbolic data, but also a set of data types specifically oriented to the representation of temporal abstractions (fig. 5).

5. Applications

The components, generated through the Tempo framework, have been integrated in different applications. They have been deployed under the form of both java libraries and web services.

5.1. Deployment as library

Libraries provide a set of API for selection and execution of the pipelines. Information can be provided by Java objects (conforming to the Tipo platform) or XML messages defined in accordance with our schema. Libraries have been integrated in a general purpose web application in which it is possible, after an authentication process, to: (a.) upload and manage data sets (b.) select, parameterize and execute one of the provided pipeline (c.) graphically visualize the results.

5.2. Deployment as web service

Web services foster the integration in distributed environments through the Simple Object Access Protocol [7] and XML messages according to the same schema mentioned in the previous section. This approach has been adopted in the Guideline Management System belonging to the Guide project [8]. In fact, for implementing computerizable clinical guidelines is often necessary to formalize sentences like: “Fever at six hours from the symptoms onset”. In order to give an overview of the pipeline functioning let’s consider a simplified sentence: “Severe Tachycardia for more than 5 minutes”, where “severe” means heart rates of 120-160 beats per minute. The pipeline defined to extract such pattern will be composed by the three modules depicted in fig. 6.

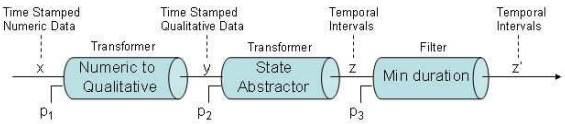

Figure 6. The pipeline generated for severe tachycardia episodes detection.

The first block transforms numeric values into qualitative ones according to the mapping (p_1) reported in Table 1. Fig. 8 depicts the heart rate time series of an

individual at rest (x) and the corresponding obtained qualitative values (y).

Heart Rate (bpm)	HR<60	60 ≤ HR<100	100 ≤ HR < 120	HR≥120
Qualitative Label	B (Bradycardia)	N (Normal)	T (Tachycardia)	S (Severe Tachycardia)

Table 1. Mapping from quantitative to qualitative values used for transforming input x into y.

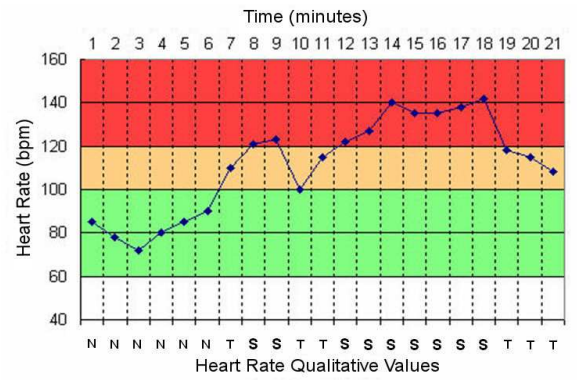

Figure 7. Heart rate time series and derived qualitative values.

The second module detects all episodes (z) of severe tachycardia through an appropriate set of parameters (p_2). The last block filters episodes giving as output only the ones (z') with a duration of at least 5 minutes (p_3). In case of a noisy time series, thanks to the pipeline architecture, a filtering block can be easily added as preprocessing step.

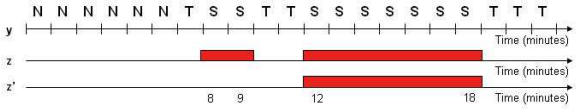

Figure 8. State abstraction and minimum duration steps of the heart rate signal data processing.

6. Conclusions and future developments

The idea of developing a framework for processing temporal data and detecting specific patterns in their course, was suggested by our past experience in developing decision support and data mining systems mainly in the medical domain. Frequently, in such application context large amounts of raw temporal data are available and it is necessary to resort to appropriate procedures for reducing the complexity of the analysis. In particular, it has been revealed very useful the application of the Temporal Abstractions technique coupled, if necessary, to filtering techniques. Given the different kind of filters and algorithms for the extraction of temporal patterns, an essential requirement has been the possibility of defining an highly configurable data analysis procedure.

In literature two similar tools have been presented: PROTEMPA [9] and KNAVE II [10]. PROTEMPA (Problem-Oriented Temporal Analysis) is a general purpose framework for detecting statistical and temporal patterns in raw clinical data, and for evaluating complex temporal relationships between the found patterns. KNAVE II is a tool that supports the visualization, summarizing, interpretation, explanation and context-sensitive navigation of time-oriented raw clinical data sets and higher-level concepts abstracted from time-oriented data.

Both frameworks are specifically oriented to clinical data processing, while Tempo is conceived as domain independent tool that is also completely independent from the source of the data and on the context. In particular, the contextualization of the abstraction procedures has been intentionally held out of the processing and is performed through a different parameterization of pipelines.

As future developments, besides adding new blocks to the default library, we are in the process of building a graphical tool for a fast definition, validation and management of processing pipelines. We planned also generator modules for feeding the pipeline with data extracted from different sources.

7. References

[1] Larizza C., Moglia A., Stefanelli M.. M-HTP: A system for monitoring heart-transplant patients. Artificial Intelligence in Medicine. 4 (1992) 111-126.

[2] Y. Shahar, A framework for knowledge-based Temporal Abstraction, Artificial Intelligence 90 (1997) 79-133.

[3] J.F. Allen, Towards a general theory of action and time, Artificial Intelligence 23 (1984) 123—154.

[4] OpenEHR [www.openehr.org]

[5] HL7 [www.hl7.org]

[6] JScience [http://www.jscience.org/]

[7] SOAP [http://www.w3.org/2000/xp/Group/]

[8] Paolo Ciccarese, Ezio Caffi, Silvana Quaglini, Mario Stefanelli. Architectures and Tools for innovative Health Information Systems: the Guide Project International Journal of Medical Informatics *,ed Marius Fieschi, Mario Stefanelli and Casimir A. Kulikowski* ,vol. 74 ,n. ,pag. 553 - 562 ,(2005)

[9] Post AR, Harrison JH. An enhanced framework for pattern detection in clinical laboratory data. Proc AMIA Symp 2002;1134.

[10] Shahar Y, Boaz D, Tahan G et al. A Web-Based System for Interactive Visualization and Exploration of Time-oriented Clinical Data and Their Abstractions. Proc AMIA Symp. 2003;1073.