journal homepage: www.intl.elsevierhealth.com/journals/cmpb

The physiology analysis system: An integrated approach for warehousing, management and analysis of time-series physiology data

*Thomas M. McKenna, Gagandeep Bawa, Kamal Kumar, Jaques Reifman**

Bioinformatics Cell, Telemedicine and Advanced Technology Research Center, U.S. Army Medical Research and Materiel Command, Fort Detrick, MD 21702, United States

ARTICLE INFO

Article history:
Received 27 June 2006
Accepted 3 January 2007

Keywords:
Physiology
Medical informatics
Data display
Time-series data analysis
Data base management system

ABSTRACT

The physiology analysis system (PAS) was developed as a resource to support the efficient warehousing, management, and analysis of physiology data, particularly, continuous time-series data that may be extensive, of variable quality, and distributed across many files. The PAS incorporates time-series data collected by many types of data-acquisition devices, and it is designed to free users from data management burdens. This Web-based system allows both discrete (attribute) and time-series (ordered) data to be manipulated, visualized, and analyzed via a client's Web browser. All processes occur on a server, so that the client does not have to download data or any application programs, and the PAS is independent of the client's computer operating system. The PAS contains a library of functions, written in different computer languages that the client can add to and use to perform specific data operations. Functions from the library are sequentially inserted into a function chain-based logical structure to construct sophisticated data operators from simple function building blocks, affording *ad hoc* query and analysis of time-series data. These features support advanced mining of physiology data.

1. Introduction

Advances in physiology sensors and data acquisition technology increasingly support collection of time-series data from patients in locations other than the clinic or laboratory. These data provide new opportunities to analyze the physiological state of individuals at times when such information is most valuable, both in real-time and for *post hoc* data mining. However, as the capability to collect time-series data advances, data may be collected under suboptimal conditions, such as during the monitoring of subjects engaged in various states of physical activity, or during transport of patients from a site of injury to hospital. These conditions can degrade the quality of the collected data. For example, cursory examination of physiology waveform data, such as electrocardiograms (ECG) collected during transport of patients shows that the waveforms are subject to patient movement artifacts, and yield records that exhibit intermittent periods of good- and poor-quality recordings [1]. Before time-series data can be comprehensively mined, it is necessary to efficiently warehouse, manage, and analyze the data files, which can, generally, be characterized as extensive, numerous, and of variable quality. A description of such files, based on data collected from trauma patients, will be given below.

The physiology analysis system (PAS) was designed as a research platform to facilitate the extraction of knowl-

* *Corresponding author at*: U.S. Army Medical Research and Materiel Command, MRMC/TATRC, Building 363, Miller Drive, Fort Detrick, MD 21702-5012, United States. Tel.: +1 301 619 7915; fax: +1 301 619 1983.
E-mail address: jaques.reifman@us.army.mil (J. Reifman).
0169-2607/$ – see front matter © 2007 Elsevier Ireland Ltd. All rights reserved.
doi:10.1016/j.cmpb.2007.01.003

edge from physiology time-series data. Its essential features reduce the data management task faced by investigators, while providing data analysis capabilities in an environment that promotes *ad hoc* queries of the databases. The system includes: (1) a platform to integrate storage, query and analysis of attribute (i.e., discrete) and time-series data, (2) a structure in which all system operations are executed via the Internet, and (3) an interface and logical structure that is as flexible and user-friendly as possible. The PAS requires only that clients have Internet access and a Web-browser; all data storage, access, analysis, and graphics functions reside on a server. Besides query and analysis, the client can visualize all attribute and time-series data, output results in reports, export the results or raw data as files, and share data and analytic routines amongst other users.

Ultimately, it is expected that the system will be used to develop knowledge that can be incorporated into algorithms that will, for instance, continuously monitor physiology variables during transport of a patient and provide a continuous read-out of the physiologic stress the patient is experiencing. A similar application is to use the mining-derived knowledge to synthesize algorithms that will allow diagnosis of injury type and severity, and prognosis of outcome based on real-time physiology data inputs from vital sign monitors, as commonly used during transport of injured civilians.

2. Data specification

The PAS is independent of specific physiology data acquisition devices; it can incorporate data collected by any device, as long as the time-series data can be expressed as ASCII files. The PAS hosts multiple physiology databases in a common interface, and it currently incorporates three databases: (1) data collected during transport of patients from the site at which they were injured to a hospital [2], (2) data collected from subjects while they were engaged in varying intensities of physical activity [3], and (3) data collected from subjects during transport on an instrumented platform similar to a litter [4]. The time-series data were collected by a variety of data acquisition devices, including the Propaq Encore, Model 206-EL (Welch Allyn, Skaneateles Falls, NY), Lifeshirt (VivoMetrics, Ventura, CA), Schiller Cardiovit AT-6 (Schiller Inc., Baar, Switzerland), SensorMedics Model 2900 (SensorMedics, Yorba Linda, CA), and a respiratory flow track board (Novametrics Medical Systems, Wallingford, CT).

A data mining example, drawn from the trauma patient database, will be used throughout this paper to illustrate features of the PAS. This database consists of attribute and time-series vital signs data collected from approximately 900 patients during helicopter transport from the site of injury to a Level-1 trauma center at the University of Texas Health Science Center at Houston, TX. The attribute data include items such as patient demographic information, injury description, and treatments. There are 100 variables of this type for each patient, and these data have already been subjected to a mining exercise [2]. In addition, twelve time-series variables were collected by Propaq 206 vital sign monitors [5] during transport; these include blood pressures, ECG, pulse oximetry, respiration, end-tidal CO_2, and additional time-series data files that are derived from the original waveforms, such as heart and respiratory rates, and arterial blood oxygen saturation.

The types of data described above have data management and analysis challenges. These include: (1) variable data frequencies, which result from data acquisition and output limits specific to vital signs monitors and the algorithms used to calculate derived values. In the trauma database, time-series frequencies range from 182 Hz for the ECG waveform to 1 Hz for derived values, such as heart rates, to sub Hz for blood pressures (i.e., single-point blood pressures are measured by employing a cuff, at intervals between 2 and 10 min), (2) the sheer volume of data—the record for an average patient is approximately a half million data points (approximately 2MB), (3) both attribute and time-series data that must be analyzed at the same time, and (4) a large number of data files to manage (e.g., more than 14,500 time-series files, which consist of the original waveform data and their associated derived values, currently stored in the database).

Existing database management systems are not optimal for the management and analysis of time-series data, since they were not built with this objective [6,7]. Their storage architecture is not structured to effectively handle time-series data and simple time-series operations are poorly supported. Querying the data requires writing statements in some form of query language [7]; this is acceptable in business applications where the same queries are performed repetitively, but becomes burdensome when the database is subject to *ad hoc* queries, as is typical of medical data mining research applications.

3. Computational methods and theory

3.1. *Function chain-based data query and analysis*

The PAS works via function chains in which data are passed through distinct functions that are sequentially executed until the analytical objective is attained. For example, a simple function chain consisting of two functions could be constructed by first applying a function to data in the trauma database to select only individuals that suffer blunt injury, followed by a second function to compute heart rate from their ECG waveforms. There can be many functions to calculate heart rate, each using a different method to do so, and the user determines which one to insert into the chain.

A function chain accomplishes two objectives. The first is to select a subset of subjects from the database, i.e., a query, based on client-supplied constraints that can be applied to attribute and/or time-series data. In the example, a constraint was used to select only individuals with blunt injuries in the trauma database.

A second function chain objective is the mathematical analysis of time-series data that is associated with each subject, to yield new time-series or scalar data. In the example, the calculation of heart rate from an ECG waveform yields new heart rate time-series data that did not previously exist.

Function chain architecture, because of its inherent modularity, allows substantial flexibility in performing *ad hoc*

queries and analyses of attribute and time-series data. Attribute data can be queried in a fashion similar to 'standard' database queries, while the original time-series data, or derived (new) time-series or scalar data that result after application of functions to the original time-series data, can be both queried and analyzed. For instance, using the example above, after calculating heart rates from ECG waveform data for a set of patients (analysis), a subset of the patients can be identified that have mean heart rates greater than 100 beats per minute (query).

3.2. *Types of functions and their use in chains*

In practice, each function in a chain is configured by picking the function from a library of functions, the variable(s) it acts upon, parameters for the function, and optionally, a constraint to be applied to the function's output. Parameters supply information necessary for a function to perform the desired action; for instance, a function that calculates the percentage of values in a time-series that fall within a value range needs to have the upper and lower limits of the value range provided as parameters. A constraint, in contrast, compares the output from a function against an objective standard, e.g., calculated heart rates must average greater than 100 beats per minute (bpm). In this case, the constrained function only passes on the variable values that pass the constraint, and the corresponding ID subset for the individuals that meet the constraint. The interactions between data type (attribute or time-series), parameters, constraints, and function output result in three function applications (Fig. 1):

(1) Functions which are similar to those in a standard database or spreadsheet. These functions are applied exclusively to attribute data and always select subjects that meet user supplied constraints (i.e., they produce a reduced set of subjects while the variable remains unchanged, Fig. 1a). Examples of such functions are ones that select subjects (IDs) based on injury type, gender, or whether they received more than 1 unit of blood in a hospital.
(2) Functions that always mathematically manipulate time-series data to generate new, derived time-series data (Fig. 1b). These functions only generate derived data; the set of subjects remains the same. An example of such a function is one that extracts a specified range of data from time-series files or that calculates a heart rate time-series from ECG waveforms. In these cases, new time-series that are smaller than the original ones are generated for each subject.
(3) Functions that always mathematically manipulate scalar or time-series data to generate new scalar data; in addition, if the user applies constraint criteria to the output of the function, the function also selects a subset of subjects. This type of function may be viewed as a hybrid between the first and second functions described above; it generates new data and can change the set of subjects (Fig. 1c). For example, this kind of function can calculate the length of time-series data files for each subject, in which case the lengths of the time-series files are generated as new data, but the number of subjects is unchanged. However, if the user specifies a minimal length constraint, then the output from this step will be the time-series file lengths for

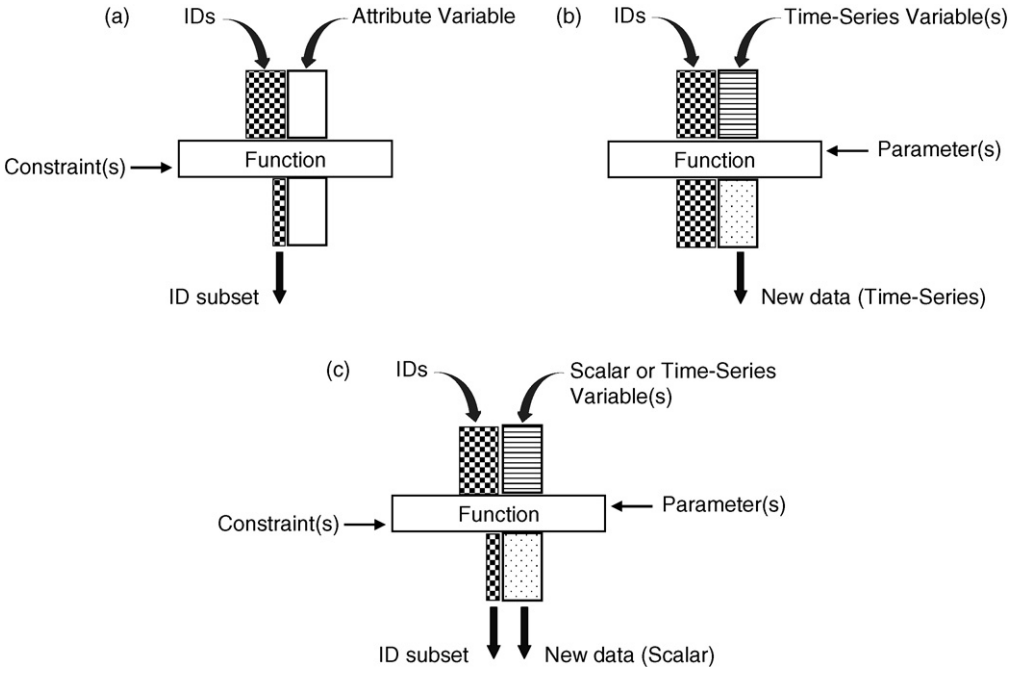

Fig. 1 – Function applications. Application (a) always applies a constraint on attribute variables (open box) to select a subset of the original subjects (checkerboard box); the attribute variable is not changed. Application (b) always takes as input time-series data (horizontal lines) to generate new time-series output (speckled box); the set of subjects is not changed. Application (c) always generates new scalar data from input scalar or time-series data; in addition, a constraint may also be applied to the derived data to select a subset of the original subjects.

only the subjects that meet the length constraint (i.e., both new data and a subset of subjects are generated).

3.3. *Function interactions*

Because function chains pass information sequentially from a preceding function to the next one in the chain, the PAS transparently ensures that only functions that can process the output from the previous function can be added within or at the end of a chain. Any function that meets the basic requirement for meshing of inter-function data exchange can be incorporated into a chain.

An important feature of the PAS is that it supports functions that process multivariate inputs. As a result, because functions can accept as input newly generated values from previously executed functions, different function chains can be assembled to progressively 'feed' their analytic output as input into other chains. In essence, instead of each element in a function chain being a function, elements in this type of function chain are function chains in their own right.

3.4. *Function library*

The PAS currently incorporates a library of over 60 functions that embody operations that can be applied to attribute or time-series data. The functions can be written in MATLAB, C, C++, Java, or FORTRAN, and the analytic objective of any function is limited only by the imagination of the investigator that codes the function. Any function placed in the PAS library is available to all users of the system. The system includes general purpose functions that can process a wide variety of time-series variables by, for example, segmenting or concatenating them, applying filters, extracting features, or performing mathematical manipulations, such as difference, max, min, and other statistical analyses. However, narrow purpose functions, which have very limited or specific inputs or outputs, are also incorporated; examples of this type are algorithms that calculate a Glasgow coma score (a measure of head injury) from certain attribute physiologic variables, or perform a power spectrum analysis on ECG time-series data. Some of the functions analyze data on a per-subject basis, e.g., calculate the standard deviation of heart rate for each patient; others act in an across-subject fashion, e.g., compute the mean value of the heart rate standard deviation for a set of selected patients.

3.5. *Boolean query*

Output at each step in a function chain where subjects are selected (i.e., a constraint is applied to decrease the set of subjects) is controlled by Boolean logic. All functions in a chain, where selection occurs, execute in a default Boolean 'AND' fashion. However, each step where selection occurs can alternatively be executed according to Boolean 'AND', 'OR', or 'NOT' operators, which can be parenthetically ordered.

3.6. *Optimized execution*

The function chain-based logical structure of the PAS allows the construction of very complex data analyses from simple function-based building blocks. Function chain execution and the evaluation of Boolean relationships are therefore optimized, by algorithms developed in-house, to minimize computational load. Furthermore, individual function results are 'remembered' during execution of a function chain so that they do not have to be recalculated if the function chain is partially modified and re-executed. To minimize storage burden, only the function chains are stored from session to session; the chains are re-executed to generate the derived values from the individual functions, when necessary.

4. System description

The user interface of the PAS is composed of several Web pages with links to switch between them. The pages have a simple, consistent HyperText Markup Language (HTML)-based interface for selecting data and entering parameters. This plain interface is used to allow the system to be accessed by different kinds of client Web browsers or computer operating systems; in this configuration no downloads of any kind are necessary to use the PAS. All attribute or time-series variables and the functions that operate on them are selected from drop-down menus; parameters or constraints on the functions are selected from drop-down menus or typed in text boxes.

The system work flow is shown as a schematic in Fig. 2. In general, the program flows from an initial login and database

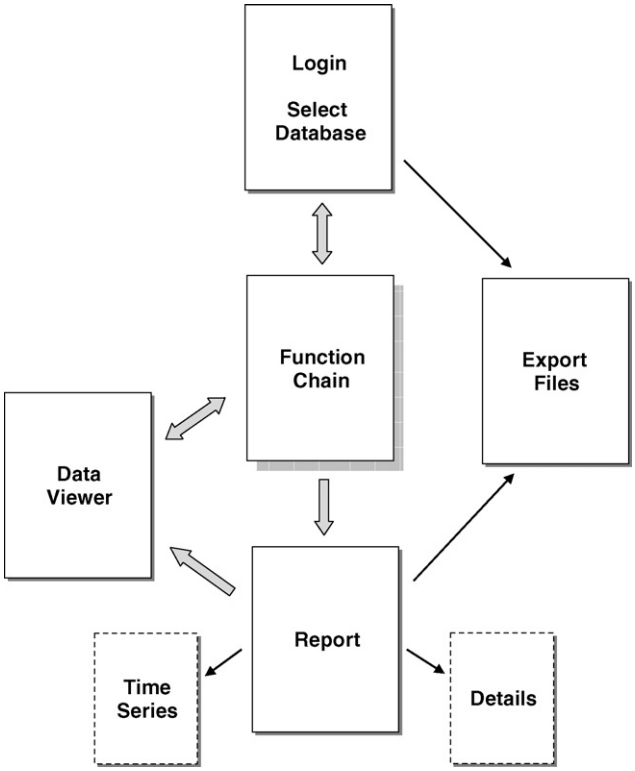

Fig. 2 – The PAS work flow. Flow follows the broad arrows, i.e., login, select database, assemble and execute a function chain, view results in a report page, and view time-series results graphically in the data viewer. The export files page is used for exporting time-series files to the client's computer for out-of-PAS analysis of the data. Two pop-up pages (dotted) display additional information about time-series or attribute data for a subject.

selection page to one in which the data analysis is set up and run, then to a report page that shows the results of the analysis and ultimately to a data viewer page to display time-series data graphically.

An optional page allows the user to export time-series files in the PAS databases from the server to the client. A primary feature of the PAS is its ability to identify a subset of subjects, and their associated time-series data files that are appropriate for advanced data mining procedures. Some of these analyses will require computational capabilities that cannot reasonably be incorporated into the PAS as a set of functions; hence the user can select and export any time-series files in the databases; in a special case, this capability is linked with a report page so that any newly created time-series data associated with the subset of subjects may also be exported. The user may select text or XML (Extensible Markup Language) as the output file format.

The PAS automatically saves function chains each time they are executed. This precludes the need to actively save function chains as they are updated as the user changes functions, variables, parameters, or constraints. The function chains persist from session to session, and they can be shared amongst users by transferring copies to other users' directories. The PAS incorporates links to an extensive help page that describes all variables, and functions and their parameters. The help page also details the experimental protocol by which data were collected for each database in the system.

5. Example

Suppose one wanted to identify patients who may suffer occult internal bleeding during transport to the hospital. A subset of patients would be searched for with the following features, which may hypothetically be associated with occult internal bleeding (Fig. 3):

(a) From patients injured by all means (penetrating, blunt, and unknown), find those with blunt injuries (Fig. 3a).
(b) Find patients with a narrow pulse pressure that falls between 5 and 20 mmHg (pulse pressure is calculated as systolic blood pressure (SBP) minus diastolic blood pressure (DBP) and should normally be about 40 mmHg, Fig. 3b).
(c) Find patients with an elevated heart rate that is greater than 100 bpm for at least 20 s (Fig. 3c).
(d) From all of the patients above (a–c), find those that first exhibit a diminished pulse pressure and simultaneously express an elevated heart rate within 300–600 s after starting transport to the hospital (Fig. 3d).

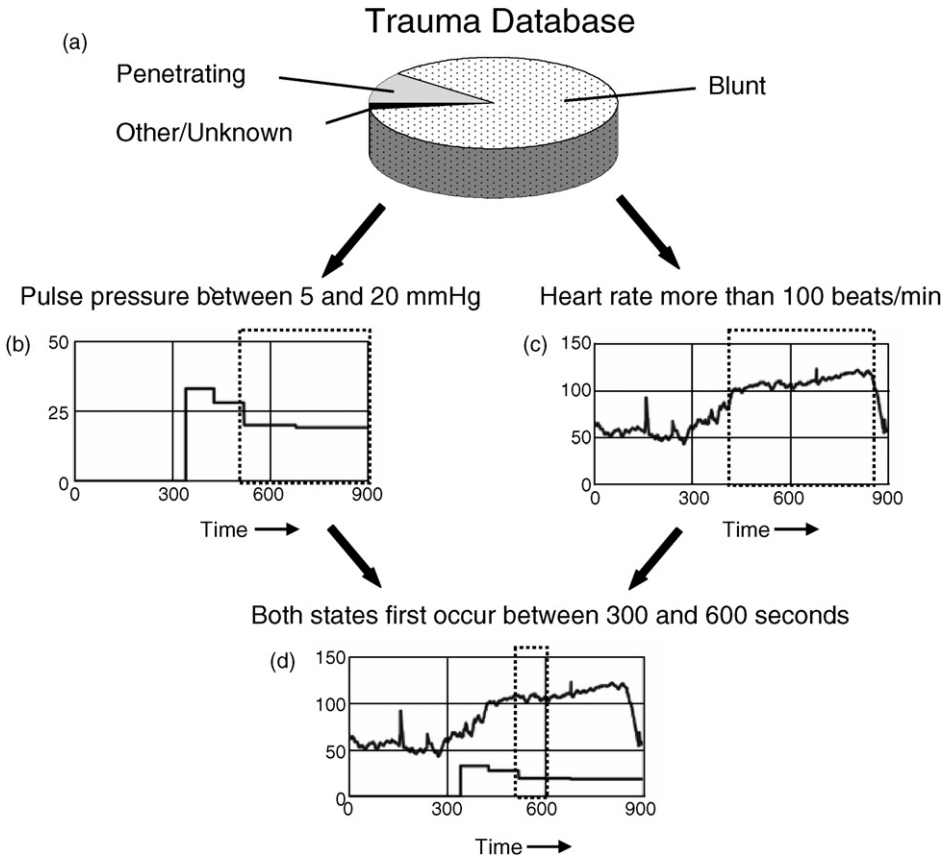

Fig. 3 – Schematic of the example query and analysis. All patients with a blunt injury (a) are selected. The subsequent panels (b–d) show results for a single patient from this set that exhibits a pulse pressure between 5 and 20 mmHg (b, dotted rectangle), and a heart rate greater than 100 bpm (c, dotted rectangle). The query and analysis is finished in (d), where the patient is identified as simultaneously meeting all of the criteria within a 300–600 s window after the start of transport (dotted rectangle).

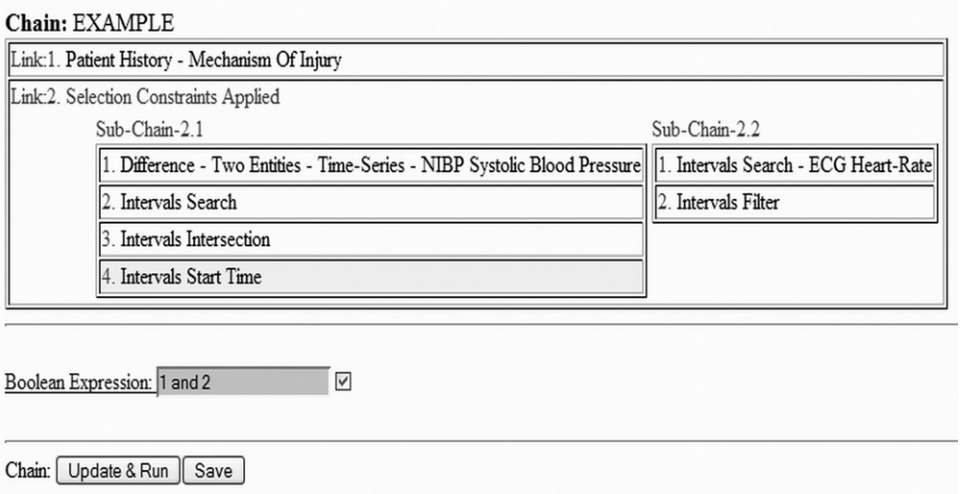

Fig. 4 – A screenshot of the example function chain. Each link indicates where a selection of a subset of patients occurs, i.e., the patient population is reduced. Link 1 incorporates a single function to select patients that suffered blunt injury. However, link 2 consists of six functions; the first five functions in this link manipulate time-series data but do not select a subset of patients, while the sixth function ("Intervals Start Time") has a constraint applied to it that allows it to select the final subset of patients. The time-series data output from the "Intervals Filter" function in sub-chain 2.2 'feeds' into the "Intervals Intersection" function in sub-chain 2.1. Data entry boxes for entering variables, parameters, and constraints pop up for each function when it is clicked, but these are not shown for clarity. The Boolean relationship between links can be modified by entering the link numbers and operators in the Boolean Expression box; in this example the default 'AND' relationship is shown.

5.1. *Function page*

Fig. 4 shows a screenshot of a function chain to identify the patients of interest. The chain consists of seven functions, which are distributed into two logical groupings termed links. Each link denotes a step in the function chain at which a selection occurs, i.e., the patient population is changed. In the example, a starting population of 898 patients is reduced to 778 patients at link 1, where only those with blunt trauma are selected, and finally to 9 patients at the end of link 2, where only the patients meeting the requisite time-series features are selected. Although link 2 contains six functions; the first five functions only manipulate time-series data but do not select a subset of patients, while the sixth function ("Intervals Start Time") manipulates time-series data and also has a constraint applied that allows it to select the final subset of patients. The "Intervals Intersection" function accepts multivariate input, where the output from the first two functions in sub-chain 2.1 and the output from 2.2 feed into it. Boxes for entering variables, parameters, and constraints pop up for each function when it is clicked, but these are not shown in the figure for clarity. The flow of data through the function chain is shown in Fig. 5.

5.2. *Report page*

A report page, which shows the output after executing the function chain, is shown in Fig. 6. Report pages consist of two parts; an upper portion in which the functions are listed, complete with all of their parameters and constraints, and a lower portion in which results are displayed in a format similar to a spreadsheet, in which each row is a subject, and the columns include the IDs of the selected subjects, their attribute or scalar data, and hyperlinks to their associated time-series data. Three other pages can be opened from the report page to provide specialized views of the results. Report pages are dynamic and incorporate multiple features:

(1) A click on the 'Detail' hyperlink for any subject opens a new page that displays all of their attribute variables in a tabular format (100 in the case of the trauma database).
(2) A click on a 'patient_id' number opens a data viewer page (see below) where all of that subject's time-series variables, both original and any newly generated by functions in the function chain, can be viewed graphically.
(3) The remaining columns display the associated attribute or scalar values, which met the constraints that were used to select the subjects. If the final output of a function chain is a time-series instead of an attribute or scalar value, then the column will display a 'time-series data' hyperlink for each subject. Clicking the hyperlink will open a page that shows all of the time-series data output from the last function for that subject.
(4) An 'Add Variable' button allows the client to add variables that were not used by the function chain to the report, if desired.
(5) Checkboxes and the 'Refresh' and 'Original' data buttons allow the client to manually reduce or restore the subject subset.
(6) An 'Export Report' button allows the report page to be exported to the client's computer as a comma separated

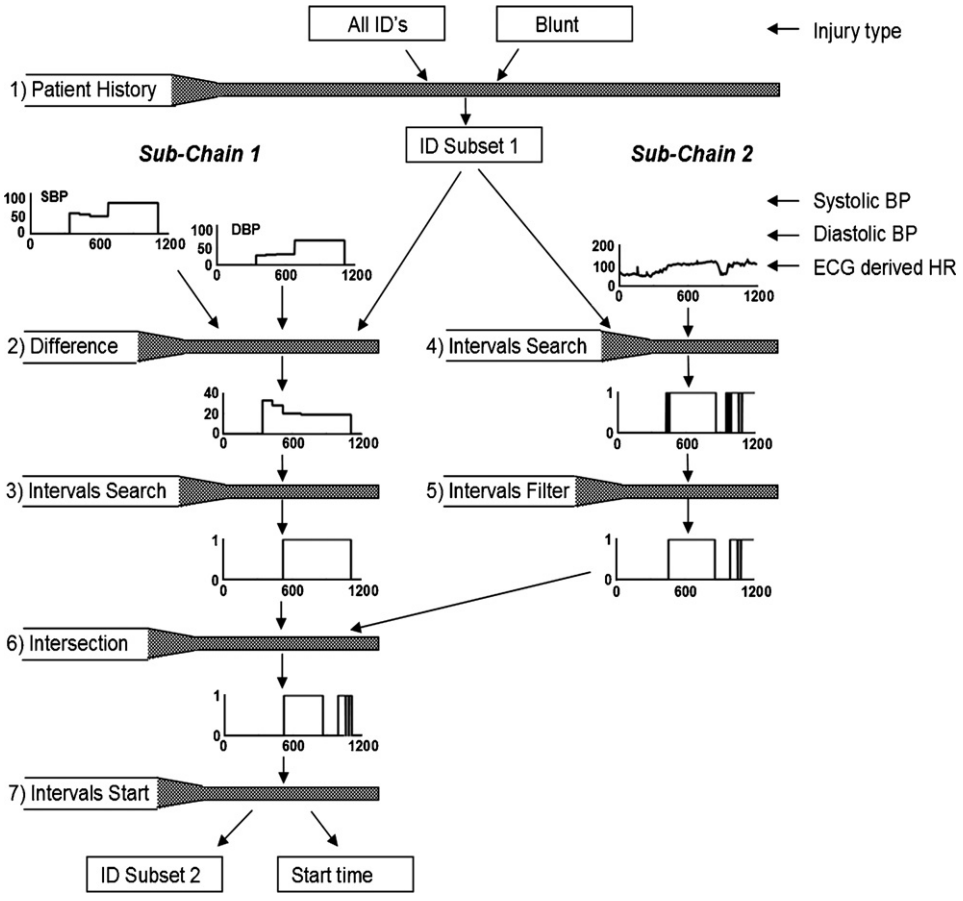

Fig. 5 – A graphical representation of function, variable, and data flow within a function chain designed to fulfill the example query and analysis shown in Figs. 3 and 4. The chain is composed of seven functions (numbered), and small graphs that show the input into and output from each function. The original input variables are listed on the right margin (injury type, systolic blood pressure, SBP; diastolic blood pressure, DBP; ECG derived heart rate, ECG HR). (1) (Patient History) is an application "a" function (Fig. 1a) that accesses the injury type attribute data and selects (passes on) only the subset of IDs from patients that were injured by blunt injury. (2) (Difference) is an application "b" function (Fig. 1b) that takes two original time-series variables, in this case systolic and diastolic blood pressures (units are mmHg, and time in s), and generates (outputs) a new time-series by subtracting one from the other on a point-by-point basis. The patient ID subset is not changed. (3) (Intervals Search) is an application "b" function that takes the output from the previous function and generates a new time-series in which difference values that are greater than or equal to 5 and less than or equal to 20 are identified (output from this and subsequent functions are coded 1 if values fall within the parameters, zero otherwise). (4) (Intervals Search) accesses original ECG derived heart rate time-series data for patients identified by the preceding Patient History function and identifies heart rates greater than 100 bpm. (5) (Intervals Filter) is an application "b" function that takes the output from the previous function and generates a new time-series that identifies intervals that are at least 20 s long. (6) (Intervals Intersection) is an application "b" function that takes the outputs from functions (3) and (5) and generates a new time-series that identifies intervals where overlaps (i.e., 1 values) occur between the time-series from both functions. (7) (Intervals Start Time) is an application "c" function (Fig. 1c) that takes the output from the preceding function, identifies the times at which the met conditions (i.e., 1 values) occur and then applies a constraint that the first identified interval must start between 300 and 600 s after the onset of data collection. The output from this function is a list of patient IDs that met this constraint and a scalar value for each of these patients that give the time when the constraint was met.

values file that can be opened in standard spreadsheet programs for out-of-PAS analysis of the results.

(7) An 'Export Time-Series' button allows the client to select any of the original or newly derived times-series variables, for any or all of the subject ID set, and to export them to their computer as text or XML files for use in other programs.

(8) A 'Save ID List' button will save the selected subset of patient IDs for use in different function chains.

5.3. *Data viewer page*

All time-series data in the system, including original and any new time-series data calculated by functions can be exam-

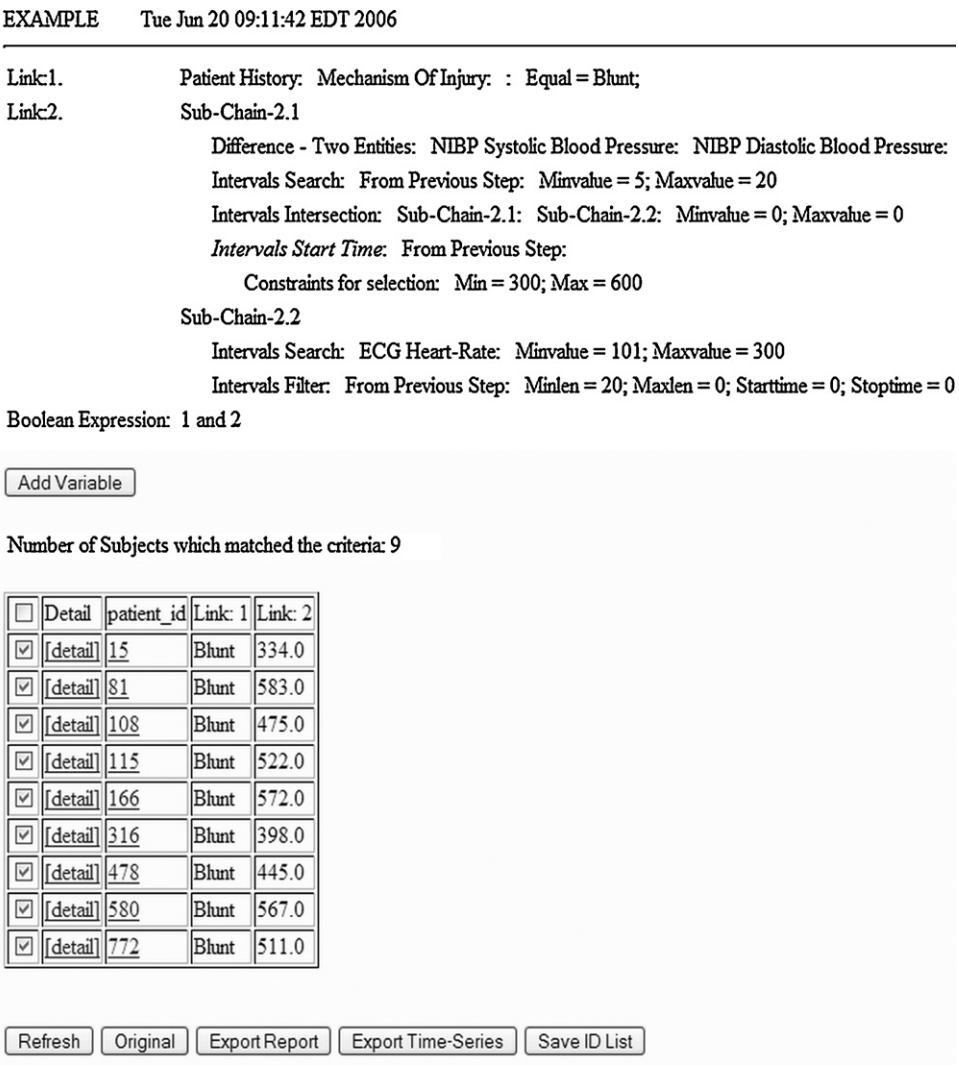

Fig. 6 – A screenshot of the report page generated after execution of the example function chain in Figs. 4 and 5. The top portion of the figure lists the functions that comprise the chain, along with their associated parameters and constraints. The table at the bottom of the report shows the result from executing the function chain; the first column (checkboxes) allows removal of individual subjects from the table, if desired, the 'detail' link in the second column opens a page showing all attribute data for a subject, the third column lists the IDs of the selected subjects (clicking on the ID number will open the data viewer to view time-series data for the subject), the fourth column (Link: 1) shows the attribute data which was used in the selection of the subset of subjects, while the fifth column (Link: 2) shows the start of the time interval where the constraints in the function terminating this link are met. In this case, all of the selected subjects suffered blunt injury (column four) and had intervals during which their heart rate was greater than 100 bpm for at least 20 s, their pulse pressure fell to between 5 and 20 mmHg, and the two intervals became coincident between 300 and 600 s after the start of transport to the hospital (column five).

ined in the data viewer. All of the small time-series graphs in Figs. 3 and 5 are screenshots from the data viewer. All standard graphics capabilities are available, including the capability to expand or decrease the size of the graph, set upper and lower ranges of the ordinate and abscissa axes, set line thickness, style, color, and markers. Time-series for multiple subjects and their associated variables can be plotted in one graph; up to nine different graphs can be shown at the same time on the page. All graphics rendering occurs on the server by a MATLAB engine (see below). Image display requires only a small automatic download of PNG (portable network graphics) files to the client's data viewer page. It is not necessary to install any viewer program or browser plug-ins in the client computer, any of the common browsers can be used, and graphics display does not depend on the power of the client's computer.

6. Hardware and software specifications

The PAS is constructed from open-source software around a MATLAB [8] computational core, and is hosted on a Dell PowerEdge 2650 server that is configured with two 2.8 GHz, 512KB

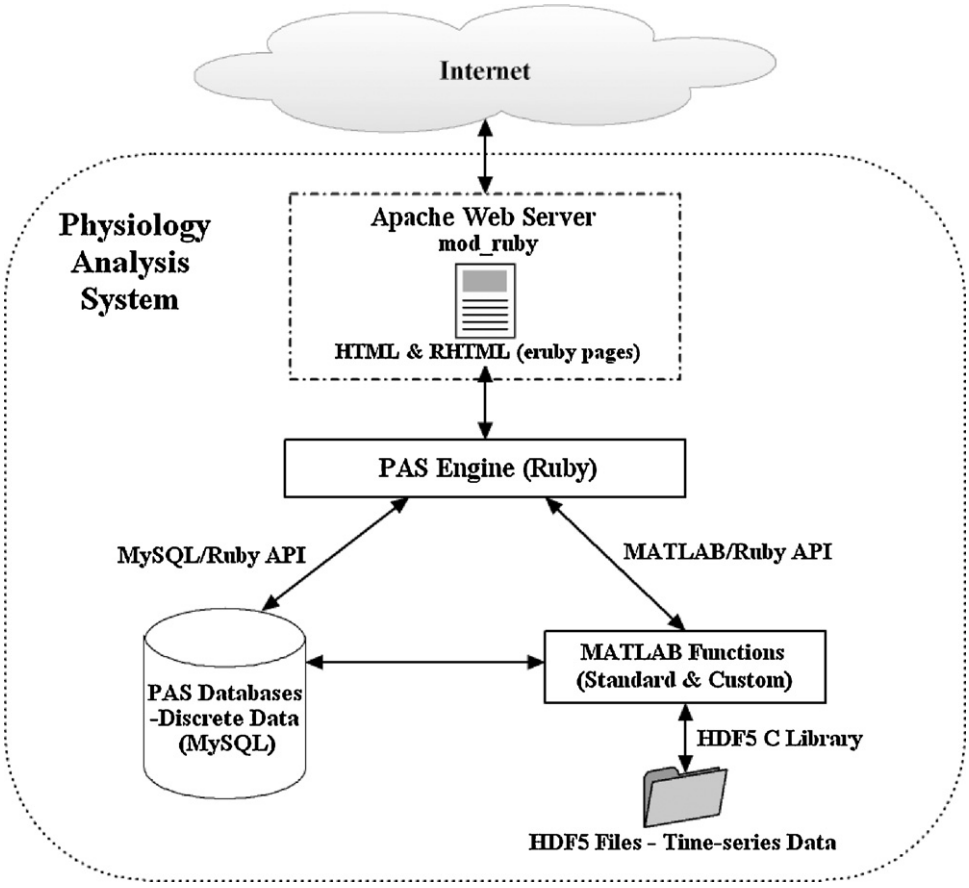

Fig. 7 – Architecture of the physiology analysis system.

Cache Intel Xeon Processors in a hyper-threaded configuration, 4GB of double-data-rate memory, a 300GB hard drive, and running the Slackware [9] 10.0 distribution of Linux. It is a server based system (Fig. 7), and like any Web application, requires a Web-server to host it. We use Apache HTTP Server [10] for this purpose. Apache receives and serves the requests for HTML and RHTML (Ruby-embedded HTML files) pages from the client's browser. RHTML pages are codes written using 'eruby' [11], a language that allows developers to embed Ruby scripts in HTML pages. Ruby [12] is the scripting language used in PAS. In order to speed up the invocation of the embedded Ruby scripts, Apache uses a tool called 'mod_ruby' [13] that embeds the Ruby interpreter into Apache, thereby allowing the scripts to be executed natively. Depending on the processing involved, these Ruby scripts or other Ruby codes invoked by these scripts may interact with:

(1) A MySQL database [14] using MySQL/Ruby [15], which is the MySQL application programming interface (API) module for Ruby. It allows Ruby to talk to MySQL databases. MySQL provides a full relational database functionality. While relational databases are not very good at managing time-series data, they are optimized for attribute data; the open source MySQL relational database is used to manage the attribute data within the PAS.
(2) MATLAB via a MATLAB API for Ruby (termed MATLAB/Ruby) that was developed in-house. It was developed using the C interface provided by MATLAB and a software development tool termed SWIG [16] (Simplified Wrapper & Interface Generator) that allows us to connect programs written in C and C++ with Ruby. MATLAB is a commercial software package that serves as the numeric computational engine in PAS to execute both standard and custom-programmed MATLAB functions; MATLAB also serves as the graphics engine. Depending on the requirement, MATLAB accesses MySQL for attribute data using the inherent API, or accesses HDF5 (Hierarchical Data Format 5) files for time-series data. HDF5 [17] is a file structure with a highly portable and extensible data format, and an associated software library that is used to store, access and manage complex data. All of the time-series data are stored in these files, because the HDF5 file structure allows a faster retrieval of time-series data, compared to relational database management systems (DBMS). For instance, the time required to load 990,000 rows of ECG time-series waveforms (approximately 218,000 data points per row) from a MATLAB data file into an Oracle 9i Release 2 DBMS Index Organized Table, versus into a HDF5 file is 37 s versus 1 s, respectively. The reverse operation, extracting 500,000 rows worth of the ECG waveform data into MATLAB from the Oracle DBMS versus the HDF5 file, required 147 s versus 0.11 s, respectively. To further increase the speed of the system, we have developed C programs to interact with HDF5 files using the HDF5 C library, and

compiled them using MATLAB to create MEX files (i.e., MATLAB Executable files; these files can be run from within MATLAB in the same way as MATLAB built-in functions), thereby allowing us to access the HDF5 files directly from MATLAB.

7. Availability

The human physiology and other medical data are collected by organizations external to our research group, the Bioinformatics Cell (BIC), and uploaded to the PAS server in batch mode. The BIC only accepts human data that have been stripped of identifiers in accordance with Health Insurance Portability and Accountability Act (HIPAA) requirements and, as such, these data are not further subject to HIPAA. The data are stored in multiple databases on a redundant array of independent disks, on a firewalled, secure shell-enabled, and user-name- and password-protected server. Sharing of retrospectively and prospectively collected data between the organizations collecting the data, the BIC, and third parties is performed under appropriate data sharing agreements.

Our objective is to use the PAS as a tool to advance the ability to develop algorithms to accurately diagnose a patient's physiological state. Within the regulatory constraints that pertain to the use and sharing of human data, the BIC will endeavor to make the PAS available as a data repository and analysis resource for military relevant medical research applications.

8. Discussion

The PAS was designed as a turn-key system to warehouse, manage, and analyze physiology time-series and attribute data. A primary design objective was to balance the conflicting demands of functionality and simplicity. Many features of the system, such as the supervision of interactions between functions, and data management and access, are transparent to the client. Simplicity of use follows from the server-based architecture of the system in which there is no requirement for plug-ins, a simple HTML interface (no applet downloads required), no necessity to write database queries (e.g., in SQL), and the fact that system upgrades are only performed at the server so that the latest version is always available to the client.

The PAS is not static; data may easily be added to existing or new databases, and new, user-defined functions inserted into the function library. Because each function is a unique calculation algorithm, it can be anything the user wishes to code into the function. For instance, a variable such as heart rate may be derived from time-series data by distinct algorithms that use various waveforms (ECG, pulse oximetry), and different features of the waveforms (QRS fiduciary points, baseline crossings) to calculate the heart rate. If it is necessary to calculate heart rate by yet another algorithm, then a new function can be coded to perform the desired action and inserted into the function library. User-defined functions expand the number of analytic methods that can be applied to the data, and therefore expand data mining capabilities.

A feature that is particularly important for mining time-series data is the ability to curate the data in the system. The quality of time-series data, particularly those collected in the field, is not assured. We have developed algorithms to qualify ECG, pulse oximetry, and respiration waveforms, to independently calculate waveform derived variables, such as heart and respiratory rates, and to assign a point-by-point quality value to these physiological variables [1,18]. Results from these algorithms are incorporated into the PAS as new time-series variables that can be searched in exactly the same fashion as the original raw data files. This allows the client to select only the best information for data mining, even if the data are plagued by periods of poor fidelity. For instance, based on our data qualification algorithms, we find that only 48%, 30%, and 24%, respectively, of ECG, pulse oximetry, and respiratory waveform time-series data in the trauma database are of good quality, but it is still possible to undertake extensive data mining using only the good portions of the waveforms.

To our knowledge, no software system exists with the same capability, flexibility, and ease of use as the PAS. A cooperative project between researchers at Beth Israel Deaconess Medical Center, Harvard Medical School, Boston University, McGill University, and MIT, under the auspices of the National Center for Research Resources of the National Institutes of Health, has resulted in an exceptionally capable system (termed PhysioNet) for the banking and analysis of physiology time-series data [19]. However, before using this system, a UNIX emulator and additional software must be downloaded, installed, and configured on the client computer. Analysis of time-series data not in the PhysioNet data bank requires conversion of the data into a specific format, and storage and management of the data files always need to be done on the client's computer. Furthermore, while elegant analysis of the data can be accomplished, this system does not perform time-series queries.

A system such as the PAS supports the investigation of time-series data under a condition where the research objective is known, but many potential approaches to extract the desired knowledge from the data exist. For instance, the U.S. Army has a program termed "Warfighter Physiological Status Monitoring" that seeks to use an array of biosensors to continuously monitor responses of individual soldiers to environmental stressors, such as heat or altitude, and to improve the likelihood of survival after wounding by providing diagnostic and treatment decision tools to first responder caregivers [20]. The number of biosensors that can be used to meet these objectives are limited, due to weight and power constraints, and the optimal combination of biosensors and time-series features (e.g., those intrinsic to individual waveforms, or complex relationships between several waveforms) to be derived from the time-series data are unknown. The PAS, which supports the management of time-series data, and extraction of features from data of variable quality via a flexible query and analysis capability, can potentially accelerate the identification of useful diagnostic or prognostic features.

Ultimately, the PAS bridges informatics with physiologic data. While attribute data are part of the conventional medical record, physiology time-series data have not historically been so. The PAS combines conventional database functions with conventional physiology time-series analysis capabilities

that do pre-exist and that have been captured in the system functions. However, to truly combine these two pre-existing functionalities, a system must allow time-series data to be manipulated in a way similar to other medical record data, i.e., the system lets one treat time-series data as any other part of the medical record. The PAS provides such a capability. This utility offers the possibility that voluminous time-series data and complex analyses could increasingly become part of the medical record itself.

Disclaimer

The opinions or assertions contained herein are the private views of the authors and are not to be construed as official or as reflecting the views of the U.S. Army or the U.S. Department of Defense.

Acknowledgements

The work was partially supported by the Combat Casualty Care and the Military Operational Medicine Directorates of the U.S. Army Medical Research and Materiel Command, Fort Detrick, Maryland. We are grateful to the University of Texas Health Science Center, and to COL John Holcomb and Dr. Jose Salinas, of the U.S. Army Institute of Surgical Research, for providing access to the Vital Signs (Trauma) database, COL Beau Freund and Dr. Beth Beidleman, of the U.S. Army Institute of Environmental Medicine, for the human activity database, and Dr. Frederick Pearce, of the Walter Reed Army Institute of Research, for the automated litter data. We thank Dr. Andrew Reisner, of the Massachusetts General Hospital Emergency Department, for providing timely and constructive feedback, and Drs. Liangyou Chen, Jingyu Liu, and Chenggang Yu from the Bioinformatics Cell for developing functions for the system.

REFERENCES

[1] C. Yu, Z. Liu, T.M. McKenna, A.T. Reisner, J. Reifman, A method for automatic identification of reliable heart rates calculated from ECG and PPG waveforms, J. Am. Med. Inform. Assoc. 13 (2006) 309–320.

[2] J.B. Holcomb, S.E. Niles, C.C. Miller, D. Hinds, J.H. Duke, F.A. Moore, Prehospital physiologic data and lifesaving interventions in trauma patients, Mil. Med. 170 (2005) 7–13.

[3] B.A. Beidleman, W.J. Tharion, M.J. Buller, R.W. Hoyt, B.J. Freund, Reliability and validity of devices for a life sign detection system, United States Army Research Institute of Environmental Medicine (Technical Report T-05-01), Natick, MA, 2004.

[4] K. Johnson, F. Pearce, D. Westenskow, L.L. Ogden, S. Farnsworth, S. Peterson, J. White, T. Slade, Clinical evaluation of the life support for trauma and transport (LSTAT) platform, Crit. Care 6 (2002) 439–446.

[5] Propaq Encore Reference Guide, Welch Allyn Inc., Beaverton, OR, 1998, URL: http://www.monitoring.welchallyn.com/products/portable/propaqencore.asp, last accessed: 2006.

[6] Oracle's Discovery Platform for Life Sciences, An Oracle White Paper, 2002, URL: http://www.oracle.com/technology/industries/life_sciences/pdf/discovery_platform_wp.pdf, last accessed: 2006.

[7] Oracle data mining 10g release 2, an Oracle white paper, 2005, URL: http://www.oracle.com/technology/products/bi/odm/pdf/bwp_db_odm_10gr2_0905.pdf, last accessed: 2006.

[8] MATLAB Web Server Version 6.5.2 Release-13, The Mathworks Inc., Natick, MA.

[9] Linux—Slackware: Version 10.0, The Slackware Linux Project, 2004, URL: http://www.slackware.com/, last accessed: 2006.

[10] Apache HTTP Server Version 1.3, HTTP Server Project, Reference Manual, URL: http://httpd.apache.org/docs/1.3/, last accessed: 2006.

[11] Eruby Version 1.0.5, Apache/Ruby integration project, URL: http://www.modruby.net/en/index.rbx/eruby/download.html last accessed: 2006.

[12] D. Kaustub, D. Grimes, A comparison of object oriented scripting languages: Python and Ruby, December 2001, URL: http://www.cs.washington.edu/homes/kd/courses/pythonruby.pdf, last accessed: 2006.

[13] Mod_ruby Version 1.2.4, Apache/Ruby integration project, September 2004, URL: http://www.modruby.net/en/, last accessed: 2006.

[14] MySQL Version 4.0, MySQL 3.23, 4.0, 4.1 Reference Manual (2006-01-11 (revision: 769)), URL: http://dev.mysql.com/doc/refman/4.1/en/, last accessed: 2006.

[15] MySQL/Ruby Version 2.5.2, URL: http://www.tmtm.org/en/mysql/ruby/, last accessed: 2006.

[16] SWIG Version 1.3.24, SWIG-1.3 Development Documentation, 2004, URL: http://www.swig.org/Doc1.3/, last accessed: 2006.

[17] HDF5 User's Guide: Version 5, National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign, Lawrence Livermore National Laboratory, Sandia National Laboratories, Los Alamos National Laboratory, Jean-loup Gailly and Mark Adler (gzip library), 2005. URL: http://hdf.ncsa.uiuc.edu/HDF5/doc/UG/index.html.

[18] L. Chen, T. McKenna, A. Reisner, J. Reifman, Algorithms to qualify respiratory data collected during transport of trauma patients, Physiol. Meas. 27 (2006) 797–816.

[19] A.L. Goldberger, L.A.N. Amaral, L. Glass, J.M. Hausdorff, P.Ch. Ivanov, R.G. Mark, J.E. Mietus, et al., PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals, Circulation 101 (23) (2000) e215–e220 (Circulation Electronic Pages; http://circ.ahajournals.org/cgi/content/full/101/23/e215).

[20] R.W. Hoyt, J. Reifman, T.S. Coster, M.J. Buller, Combat medical informatics: present and future, Proc. AMIA (2002) 335–339.