BEARIFY REQUIREMENTS SPECIFICATION

PROJECT INFORMATION	
Project Name:	Bearify
Project Manager:	Erik Gregg

DOCUMENT INFORMATION	
Subject:	Requirements specification
Authors:	Timo Ahonen
Keywords:	
Comments:	
Creation date:	25.1.2002
Revision date:	2.2.2002
Document Status:	[ ] Draft
	[ ] Proposal
	[X] Inspected & approved
Revision History:	25.1.2002 Version 0.1 Original draft by TA
	27.1.2002 Version 0.2 Improved draft by TA after discussion with Sauli Kivikunnas
	28.1.2002 Version 1.0 Proposal for inspection meeting by TA
	2.2.2002 Version 1.1 Corrected after inspection meeting by TA
	6.2.2002 Version 1.1 Correnctions approved by TR

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to specify the requirements of the Dynamic Time Warping Trend Analysis Tool, which will be created within the DTREND-project. The supplier in the project is the DTREND group and the customer is VTT Electronics.

This software project is intended to fulfill the requirements of the Advanced Software Engineering Project course (52479S) of University of Oulu.

1.2 Scope

The aim of the project is to create a pattern recognition tool for process monitoring and diagnostics to be used in VTT's research projects.

The customer may use some of the components created in the DTREND project in its own software projects. However, the DTREND project is completely independent of these other projects.

1.3 Acronyms and symbols

DLL	Dynamic Link Library.
DTW	Dynamic Time Warping.
GUI	Graphical User Interface
S	The observation vector.
s_i	An element of the observation vector S.
n	Length (dimension) of the observation vector.
T	The template vector.
t_j	An element of the template vector T.
m	Length of the template vector.
W	The warping path vector.
$w_k = (i_k, j_k)$	An element of the warping path vector W.
p	Length of the warping path vector.
K	Length of the warping path.
k	An unique index number of a template.
A	A set of all allowable warping paths.
ω	The warping window width.
δ	Warping path start and end point offset.

1.4 References

[1] IEEE 830–1998 Standard

[2] Berndt, D.J. & Clifford, J. 1996. Finding patterns in time series: A

dynamic programming approach. In: Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P. & Uthurusamy R. (eds.) Advances in Knowledge Discovery and Data Mining. Menlo Park, AAAI Press / The MIT Press. P. 229-248.

[3] Keogh, Eamonn J. & Pazzani, Michael J. 2001. Derivative Dynamic Time Warping. First SIAM International Conference on Data Mining (SDM'2001), April 5-7, Chicago, IL. USA.

[4] Paulus, D. W. R. & Hornegger, J. 1998. Applied Pattern Recognition: A Practical Introduction to Image and Speech Processing in C++. Wiesbaden, Verlag Vieweg. 405 p.

[5] Synthetic Control Chart Time Series, University of California, Irvine, USA. Last update 14.6.1998. Referenced on 27.1.2002.
<http://kdd.ics.uc.edu/databases/synthetic_control/
synthetic_control.html>

[6] Dynamic Time Warping Digit Recognizer, Institution for Signal and Information Processing, Missisippi State University, USA. Last update 18.1.2002. Referenced on 27.1.2002.
<http://www.isip.msstate.edu/projects/speech/software/demonstrations/
applets/util/dynamic_time_warping/current/index.html>

[7] Intel Recognition Primitives Library. Referenced on 27.1.2002.
<http://developer.intel.com/software/products/perflib/rpl/index.htm>

1.5 Overview

The rest of this document is divided into four parts. Section 2 contains a general introduction to the functions of the DTREND tool. Specific descriptions of the product functions are in section 3. The last two sections contain some general requirements concerning the whole project.

This document complies with IEEE Recommended Practice for Software Requirements Specifications [1].

2 General description

2.1 Project perspective

The goal of the DTREND project is to create a research tool which implements the Dynamic Time Warping algorithm. The tool will be used in research projects of VTT Electronics. These research projects are concerned with process monitoring and diagnostics. The main goal of the DTREND tool is recognition of patterns in measurement time series.

Figure 1: Parts of the DTREND software

The most important characteristic of the DTW algorithm and DTREND tool is the ability to cope with variation on the time axis, i.e. some parts of the signal may be shrunken whereas other parts may be stretched.

The DTREND tool must be divided into two parts: A function library, which implements the DTW algorithm and a executable which utilizes the function library. The executable part is called the GUI part in this document, even though strictly speaking, not all of its functions are those of a GUI. The parts of the software are visualized in Figure 1.

2.2 Product functions

2.2.1 Use cases

1. Load records from a text file.
2. Pick part(s) of a record to be used as templates or as an observation.
3. Name the templates and the observation.
4. Save the templates to a text file.
5. Save the observation to a text file.
6. Load an observation from a text file.
7. Load templates from a text file.

8. Choose the parameters of the DTW algorithm. These include
   (a) Global constraints
   (b) Local constraints
   (c) How the distance between two points is determined
9. Compare the observation against all templates using the DTW algorithm and announce, which template matches best and the distance between the observation and the template found.
10. Compare the observation against a chosen template and show the distance between them.
11. Show the warping grid and the warping path of the last run. Show also the lengths of the axes (i.e. the length of the template and the observation).
12. Draw plots of the (best-fit) template and the observation of the last DTW run.
13. Save the parameters and the results of the run to a log file.

2.2.2 The DTW algorithm

The DTW algorithm is so essential part of the software that it will be discussed here briefly. More thorough descriptions can be found in the references [2], [3] and [4].

We have two time series, S and T. Let $d(i,j)$ be the distance between points s_i and t_i. Now we try to find such path W in the $i \times j$ matrix that minimizes the quotient

$$\frac{\sum_{k=1}^{p} d(w_k)}{K} = \frac{\sum_{k=1}^{p} d(i_k, j_k)}{K} \tag{1}$$

where K is the length of the path. Now we have

$$DTW(S,T) = \min_{W} \frac{\sum_{k=1}^{p} d(w_k)}{K} \tag{2}$$

We restrict the algorithm by a local constraint so that the path may go from point (i_k, j_k) to point (i_k+1, j_k), (i_k+1, j_k+1) or (i_k, j_k+1) , i.e. we set continuity and monotonicity requiremnts.

The set of allowable paths may be restricted with parameters discussed in section 3.1.3. When parameter P_1 restricts the paths to set A_1 and simultaneously parameter P_2 to set A_2, the total set of allowable paths is the intersection of these two sets, i.e.

$$A = A_1 \cap A_2 \tag{3}$$

2.2.3 Software functions

The software functions are listed here. A complete description of these functions is in section 3.1
DLL library functions

- *D-01*: Load template
- *D-02*: Delete template
- *D-03*: One to all comparison
- *D-04*: One to one comparison
- *D-05*: Return template

GUI functions

- *G-01*: Load data set
- *G-02*: Select data
- *G-03*: Save selection
- *G-04*: Save vector to file
- *G-05*: Load vector from file
- *G-06*: Delete template from GUI
- *G-07*: Select DTW parameters
- *G-08*: Compare to all
- *G-09*: Compare to one
- *G-10*: Save results to log
- *G-11*: Show plots

DTW parameters

- *P-01*: Distance metrics
- *P-02*: Warping window
- *P-03*: Slope constraint
- *P-04*: Start and end point offset

2.3 User characteristics

First user group of the tool is research personnel, who are familiar with the DTW algorithm. This user group needs especially a logging feature.

Another user group is research personnel who use the tool in learning the method. For this user group, the tool must contain a clear GUI with possibility to plot the observation and template vectors and the warping path on the screen.

The third user group which has to be considered during the software project are other software engineers who do further development of the tool. To make further development as easy as possible, the software must be well documented.

2.4 General constraints

The tool will be run on a PC workstation running Microsoft Windows NT.

The DTW algorithm must be implemented in a Windows DLL library. The library must be well documented, because the customer may use it in its own projects.

The tool is stand-alone, i.e. there is no need for network functionality.

The amount of work reserved for the project is three man months.

The product must be ready and tested by the 15th March 2002.

2.5 Assumptions and dependencies

The non-free tools (eg. Microsoft Visual C++) needed within the project will be provided by the customer.

3 Structural overview and specific requirements

3.1 Functional requirements

In this section, the DLL library and the GUI are handled separately. The GUI must use the functions provided by the DLL. The requirements concerning the parameters of the DTW are also in a separate section.

The requirements are divided into three classes:

- Essential: Abscence or malfunction of these functions makes the software unusable, in which case the project will not be accepted.
- Conditional: These functions are not absolutely necessary, but leaving them out decreases significantly the value of the product.
- Optional: These functions may or may not be worthwhile and the supplier may propose leaving them out at design or implementation stage.

A more thorough description of these three classes can be found in reference [1].

3.1.1 The DLL library

Functional requirement D-01: Load template (Essential)

- Introduction:

  The DLL must contain a function to load templates into it.

- Input:

  A vector T of floating point values and the length of the vector.

- Processing:

  The DLL copies the values into its internal data structures to be used by functions *D-03* and *D-04*.

- Output:

  A unique index number k of the saved template.

Functional requirement D-02: Delete template (Conditional)

- Introduction:

  The DLL must contain a function to delete templates from its data structures.

- Input:

  Index number k_d of the template to be deleted

- Processing:

  The DLL deletes the template in question from its data structures.

- Output:

  None

Functional requirement D-03: One to all comparison (Conditional)

- Introduction:

  The DLL must contain a function to compare a given observation against all templates.

- Input:

  1. A vector S of single-precision floating point values containing the observation

2. The length of the vector S
3. A structure containing the parameters for the algorithm. These parameters are discussed in section 3.1.3

- Processing:

  The DLL runs the DTW algorithm between the observation and every template that the user has entered earlier.

- Output:

  A vector of vectors, which contain two values:

  1. Index number of a template
  2. The calculated DTW difference between the observation and the template in question

Functional requirement D-04: One to on comparison (Essential)

- Introduction:

  The DLL must contain a function to compare a given observation against one specific template.

- Input:

  1. A vector S of single-precision floating point values containing the observation
  2. The length of the vector S
  3. A structure containing the parameters for the algorithm. These parameters are discussed in chapter 3.1.3
  4. Index number of the template that the observation is compared against

- Processing:

  The DLL compares the observation against the selected template using the DTW algorithm with selected parameters.

- Output:

  1. The DTW difference between the observation and the template
  2. A vector W containing the best-fit warping path. The vector consists of (i, j)-pairs that define the path.
  3. Length of vector W

Functional requirement D-05: Return template (Essential)

- Introduction:

  The DLL must contain a function to return a template that has been stored in its data structures.

- Input:

  The index k of the template to be returned.

- Processing:

  The DLL returns the template in question.

- Output:

  1. The requested template, T_k.
  2. Length of T_k.

3.1.2 Functional requirements of the GUI

Functional requirement G-01: Load data set (Conditional)

- Introduction:

  The user must be able to load a data set from a text file.

- Input:

  The file is selected with a file selection dialog.

- Processing:

  The GUI shows the file selection dialog. When user has selected the file, GUI opens the file in a data selection view described in requirement *G-02*. The file format is described in section 3.2.1.

- Output:

  If the file format is invalid, the GUI shows an error message and returns to main view. Otherwise, the GUI opens the data selection view.

Functional requirement G-02: Select data (Conditional)

- Introduction:

  There must be a data selection view, which can be used to select parts of the data set. See figure 2. Note that the figure is intended only to clarify, what elements the selection view must contain.

- Input:

  The file opened at stage *G-01*.

  The selection view itself must contain following user inputs:

Figure 2: Contents of the data selection view

  - Which record of the data set is to be shown
  - Select a part of the record plot by painting
  - Select a part of the record plot by entering the indexes of the first and last element to be selected.

- Processing:

  When user selects the record to be shown, the chosen record must be read from the data set file and shown as a trend plot in the view. When user selects a part of the record by painting, the "first element" and "last element" fields are updated respectively and vice versa. The selected area is marked in reverse colours. When user chooses to save the selected part as a template or as an observation, a vector containing the values inside the selection is passed along.

- Output:

  - A plot of the chosen record and the record length.
  - A vector containing the elements inside the selection and the length of the vector.

Functional requirement G-03: Save selection (Conditional)

- Introduction:

  The user must be able to save the selected part as a template or as an observation.

- Input:

  A vector from *G-02* containing the values to be saved and the length of the vector and following user inputs:

  - Is the selection intended to be saved as a template or as an observation

  - A name for the template / observation

- Processing:

  When user chooses to save the selected part, he is asked to name it. Then the selection is saved as a template or as an observation. There may be only one observation in the memory at a time, so if the selection is saved as an observation, the previous observation is erased.

- Output:

  None

Functional requirement G-04: Save vector to file (Conditional)

- Introduction:

  The user must be able to save the observation or the templates in memory to a text file.

- Input:

  - Which data is to be saved: the observation or the templates
  - Name of the file
  - If the file exists, the user is asked whether to overwrite the file

- Processing:

  The user is shown a file selection dialog, using which he/she may choose, which file the information is saved to. If the file exists, the user is asked whether to overwrite it. If the user selects not to overwrite the file, the file name is asked again. There is no possibility to apped to files. After the file has been selected, the observation or templates including the names given to them are saved to the file. The exact file format will be determined at design stage.

- Output:

  - A message that the saving has succeeded or an error message.
  - The file containing the observation or the templates and their names

Functional requirement G-05: Load vector from file (Essential)

- Introduction:

  The user must be able to load templates and an observation directly from a text file.

- Input:

  - Which data is to be loaded: the observation or the templates
  - Name of the file

- Processing:

  The user is shown a file selection dialog. The selected file is opened and the data (name of the template/observation and the values) is read into memory. The exact file format will be determined at design stage. If the user loads an observation, the previous observation is erased. If the the user loads templates, they are added to the template list.

- Output:

  - A message to user that the loading has succeeded or an error message.
  - A list of vectors containg the loaded templates or observation

Functional requirement G-06: Delete template from GUI (Conditional)

- Introduction:

  The user must be able to delete templates from the memory (one at a time or all).

- Input:

  - Whether to delete one or all of the templates
  - The template to be selected

- Processing:

  The programs deletes the selected templates from the memory.

- Output:

  None

Functional requirement G-07: Select DTW Parameters (Essential)

- Introduction:

  The user must be able to select the parameters of the DTW algorithm

- Input:

  The chosen parameters

- Processing:

  The parameters of the algorithm are described in section 3.1.3

- Output:

  None

Functional requirement G-08: Compare to all (Conditional)

- Introduction:

  The user must be able to compare the observation against all templates in memory.

- Input:

  None

- Processing:

  The program compares the observation against all templates in memory using the DTW algorithm with parameters chosen in *G-07*.

- Output:

  - Name of the best-fitting template and the calculated DTW distance
  - A plot of the observation and the template and the warping path (see *G-11*)

Functional requirement G-09: Compare to one (Essential)

- Introduction:

  The user must be able to select one of the templates and run the DTW algorithm against it.

- Input:

  The selected template.

- Processing:

  The observation is compared against the chosen template with the DTW algorithm using the parameters selected in *G-07*.

- Output:

  - Name of the selected template and the calculated DTW distance
  - A plot of the observation and the template and the warping path (see *G-11*)

Functional requirement G-10: Save results to log (Conditional)

- Introduction:

  The user must be able to save the results of the last run to a log file.

- Input:

  The results of the run and the name of the log file as user input.

- Processing:

  The user is shown a file selection dialog. If the user chooses a file that exists already, the user is asked whether the file should be rewritten. There is no possibility to append to a file. The results are saved to the chosen file.

- Output: The thing listed here are the minimum requirements, i.e. also other things may be saved to the log file.

  1. If the last run was of type *G-08*:
     - Date and time
     - The algorithm parameters
     - Name of the observation
     - Name of every template in memory and the DTW distance between the template and the observation. The templates must be in ascending order by the DTW distance.
     - Ordered list of points w_k of the warping path of the best-fit template.
  2. If the last run was of type *G-09*:
     - Date and time
     - The algorithm parameters
     - Name of the observation
     - Name of the template and the DTW distance between the observation and the template
     - Ordered list of points w_k of the warping path.

Functional requirement G-11: Show plots (Essential)

- Introduction:

  The observation, best-fit template, the warping path and the warping window must be visible in the GUI (see Figure 3).

- Input:

  The observation, template and warping path vectors and the warping window width ω.

- Processing:

  The GUI draws plots of the vectors.

- Output:

  The plots and axis lengths as in Figure 3. The observation is under the path and the template is on the left of the path.

Figure 3: The template, observation, warping window and warping path

3.1.3 Requirements concerning the parameters of the DTW algorithm

Algorithm parameter requirement P-01: Distance metrics (Essential)

- Introduction:

  Two types of distance metrics must be supported: Euclidian and derivative metrics. The interfaces must be designed so that it is easy to add support for other metrics at a later time.

- A closer view:

  See [3].

  1. Euclidian distance metrics:

     $$d(i,j) = (s_i - t_j)^2 \tag{4}$$

  2. Derivative distance metrics:

     $$d(i,j) = (D_x[s_i] - D_x[t_j])^2, \tag{5}$$

     where $D_x[t_j]$ is an estimate of the derivative of t_j:

     $$D_x[t_j] = \frac{(t_j - t_{j-1}) + ((t_{j+1} - t_{j-1})/2)}{2}, \tag{6}$$

when $2 \leq j \leq m-1$. The estimates for first and last elements of the sequece are defined followingly:

$$\begin{aligned} D_x[t_1] &= D_x[t_2] \\ D_x[t_m] &= D_x[t_{m-1}] \end{aligned} \tag{7}$$

Algorithm parameter requirement P-02: Warping window (Essential)

- Introduction:

  The algorithm must be able to constrain the warping path inside a warping window. It must be possible to change the size of the window.

- A closer view:

  See [2], page 234. The warping path may contain only such points $w_k = (i_k, j_k)$ for which

  $$|i_k - j_k| \leq \omega \tag{8}$$

  The value of ω must be passed as a parameter to the algorithm.

Algorithm parameter requirement P-03: Slope constraint (Essential)

- Introduction:

  The algorithm must support only the slope constraint mentioned in chapter 2.2.2. It must be possible to add other slope constraints at a later time.

- A closer view:

  The warping path may go from point (i_k, j_k) to point (i_k+1, j_k), (i_k+1, j_k+1) or (i_k, j_k+1). The interfaces must be designed so, that it is possible to implement other slope constraints (eg. Itakura slope constraint, see [1]).

Algorithm parameter requirement P-04: Start and end point offset (Conditional)

- Introduction:

  The algorithm must allow using fixed start and end points or a certain offset at the start and end points of the warping path. Fixed points are used by default.

- A closer view:

  1. Fixed points:

  $$\begin{aligned} i_1 &= 1 \\ j_1 &= 1 \\ i_p &= n \\ j_p &= m. \end{aligned} \tag{9}$$

2. Slightly relaxed points:

$$\begin{array}{rcccl} 1 & \leq & i_1 & \leq & 1+\delta \\ 1 & \leq & j_1 & \leq & 1+\delta \\ n-\delta & \leq & i_p & \leq & n \\ m-\delta & \leq & j_p & \leq & m \end{array} \tag{10}$$

   The value of δ must be passed as a parameter to the algorithm.

3.1.4 Dependencies between the functional requirements

The following table shows how the requirements depend on other requirements, i.e. which requirements have to be completed in order to complete next requirement.

Requirement	**Depends on**
D-01	-
D-02	*D-01*
D-03	*D-01, P-01, P-02, P-03*
D-04	*D-01, P-01, P-02, P-03*
D-05	*D-01*
G-01	-
G-02	*G-01*
G-03	*G-02, D-01*
G-04	*G-03, D-05*
G-05	*D-01*
G-06	*D-02*
G-07	-
G-08	*D-03*
G-09	*D-04*
G-10	*D-05* and *G-08* or *G-09*
G-11	*G-08* or *G-09*
P-01	-
P-02	-
P-03	-
P-04	-

3.2 External interface requirements

3.2.1 The data set file

The data set file is pure ASCII text file. One row contains one record. There is at least one row in the file.

A record contains at least three positive or negative decimal numbers separated with one or more spaces. The values are consecutive in time, and there is no time stamps for the values. A dot is used as a decimal separator. A number doesn't necessarily have to contain a decimal separator.

Particularly, the program must be able to read the data file in reference [5].

3.2.2 The graphical user interface

The graphical user interface must contain only subdued colours. The GUI must contain VTT's logo.

3.3 Performance requirements

Since the DTREND tool is intended to be a research tool, there are quite few specific requirements. Some typical values for vector lengths are stated here.

Typical length of observation and template vectors is less than 60. Yet, the data structures must be designed so that there is no absolute limit for the lengths (apart from the physical memory available).

Typical number of templates in memory at a time is 12 and the maximum number is 40. Use of dynamic data structures is advisable here, too.

The maximum running time for one to one comparison (*G-09*) with vectors of length 60 is 10 seconds.

3.4 Design constraints

The DLL library must be implemented in C++ and the GUI in Java.

4 General requirements

4.1 Security requirements

There are no specific security requirements.

4.2 Reliability requirements

The GUI must handle following exceptions and show a message describing the error:

- Invalid user input (eg. text instead of a number)
- File not found
- No access to a file
- Invalid file format

4.3 Maintainability requirements

It is possible that the customer continues developing the software. It is important to pay attention to this during the software design and implementation.

The DTREND project group is not responsible for maintaining the software after the project.

4.4 Portability requirements

Because of the use of the DLL technology the software is rather tightly bound to Microsoft Windows environment. There is no need to consider portability in design or implementation.

4.5 Installation requirements

No installation program is needed.

4.6 Administration handling

Default configuration may be coded into the software. There is no need for further configuration handling

4.7 Diagnostic requirements

Apart from the error messages mentioned in section 4.2 there are no specific diagnostic requirements. The log file mentioned in section 3.1.2 is intended to save the results of the algorithm run, not to diagnostics.

5 Software generation and integration

Tools to be used:

- C++: Microsoft Visual C++ (will be provided by the customer)
- Java: Sun JDK 1.3, Borland JBuilder Personal
- Documents: LaTeX, LaTeX2HTML
- Version handling: CVS

Directory structure:

- Main project CVS directory:
  s-tko-pc11.oulu.fi:/home/dtrend/cvs
- Subdirectories:

- doc: document files
- bin: release binary files
- bin_debug: debug versions of binaries
- source/gui: GUI source files
- source/dll: DLL source files