Requirements Document

Relevans

Helsinki 1st March 2005

Software Engineering Project

UNIVERSITY OF HELSINKI
Department of Computer Science

Course
581260 Software Engineering Project (6 cr)

Project Group
Antti Nyberg
Päivi Suomalainen
Teppo Ahonen
Jussi Kurki

Client
Miro Lehtonen

Project Masters
Juha Taina
Antti Tevanlinna

Homepage
`http://www.cs.helsinki.fi/group/relevans`

Change Log

Version	Date	Modifications
0.1	05.02.2005	First draft with class diagram
0.2	08.02.2004	Revision with more accurate requirements

Contents

1 Introduction

ScoreCombiner is a Software Development Project on a course of the same name at the University of Helsinki. The project is implemented as a Cum Laude-level project work. All of the pertinent material will be available at the team homepage

`http://www.cs.helsinki.fi/group/relevans`

Chapter 2 gives the reader an overview of the system, as well as its connections and interfaces to other systems. Requirements imposed on the system by our clients are stated in Chapter 3 and, finally, based on those requirements, system functions are formed in Chapter 4.

2 Overview

The INEX project at the University of Helsinki is about XML Information Retrieval. As part of a competition, tools and methods for that purpose have been produced since 2002. Our two customers - familiar with the INEX project from past years, and likely participants in future ones - hope to improve their search engine's odds. They need some means of experimenting with different score-combining schemes, and this is what we are commissioned to provide.

The participants design search engines to find, for a given query, relevant answers from a large set of XML documents arranged under several topics. They themselves contribute to the set from which the used queries are *ultimately* selected. They submit from 1 to 3 sets of answers, each containing a sorted list of *up to* 1500 most relevant document fragments. The 100 highest-scoring answers from each set are judged by humans, and the rest of the answers are awarded *based* on comparison between all participants' sets of answers.

Customers: is this more or less correct?

2.1 Terminology

Some of the notation, terms and abbreviations used in this document are defined below:

XML a language for describing structured documents

collection an index into the set of XML documents arranged under several topics

query a search performed on the collection, comparable to queries performed on search engines like Google and Yahoo

relevance a measurement by which to rank potential answers to a query

score a listed value given for all MRUs based on their relevances but calculated by ScoreCombiner for other document fragments as well

size a listed value given to all document fragments based on a chosen unit of measurement - characters, words, sentences and elements being some of the possible units

document fragment a fragment of an XML document delimited by some XML element - for example, the opening and closing tags of a paragraph - including all of the content in between

MRU (Minimal Retrieval Unit) a document fragment of defined minimal size - a paragraph, subsection or a small section - for which scores are provided

RU (Retrieval Unit) see document fragment

score combination the process of scoring non-minimal RUs based on external factors like known MRU scores, the relationship or size of the involved RUs or their level of branching (i.e. number of children)

Upward Propagation a known score combination method where the score of a RU is derived from size-weighted scores of its child RUs

RSV (Retrieval Status Value) a.k.a. Ranking Status Value; a term used in the literature for the parameter of upward propagation; see relevance

nested scores the case with *some* older score lists where a score has been calculated for some RU that also has scored descendants, making the score of the parent obsolete; a potential point of comparison for the calculated scores

deprecation a Java term for features of the programming language that may not be supported by future versions

formula superclass a Java class to be implemented that enables the addition of new formulae through simple, separately compiled extensions of itself

ScoreCombiner a working title for the program

2.2 Putting things into perspective

The following list is our starting point. Here, the processing of our customers' search engine is divided into four distinct steps that are listed in natural order of succession:

1. *Query processing*: queries are processed in some way that yields *useable* information based on which the document fragments are scored

2. *Score computation for each* fragment *in the index*: MRU scores are calculated using the information derived from the processing of a query

3. *Score combination*: the scores of non-minimal RUs are calculated based on known external factors like MRU scores, the relationship or size of the involved RUs or their level of branching

4. *Listing the best answers*: 1500 highest-scoring RUs with non-overlapping content are listed in descending order

Each of these routines are currently implemented as Perl scripts. Our customers have estimated score combination to be the weakest part of their existing system, and we have been contracted to facilitate the search for, as well as use of, alternate solutions. While steps 3 and 4 of the process are distinct and the only new implementation actually needed would be *Score Combination* (3), implementation of the latter (4) is included in the project to broaden the scope of the system's capabilities. For example: saving an intermediary result would not be meaningful in this context.

Now, the problem domain can be visualized, and Figure 1 does just that. The part to be implemented is highlighted in different color for clarity.

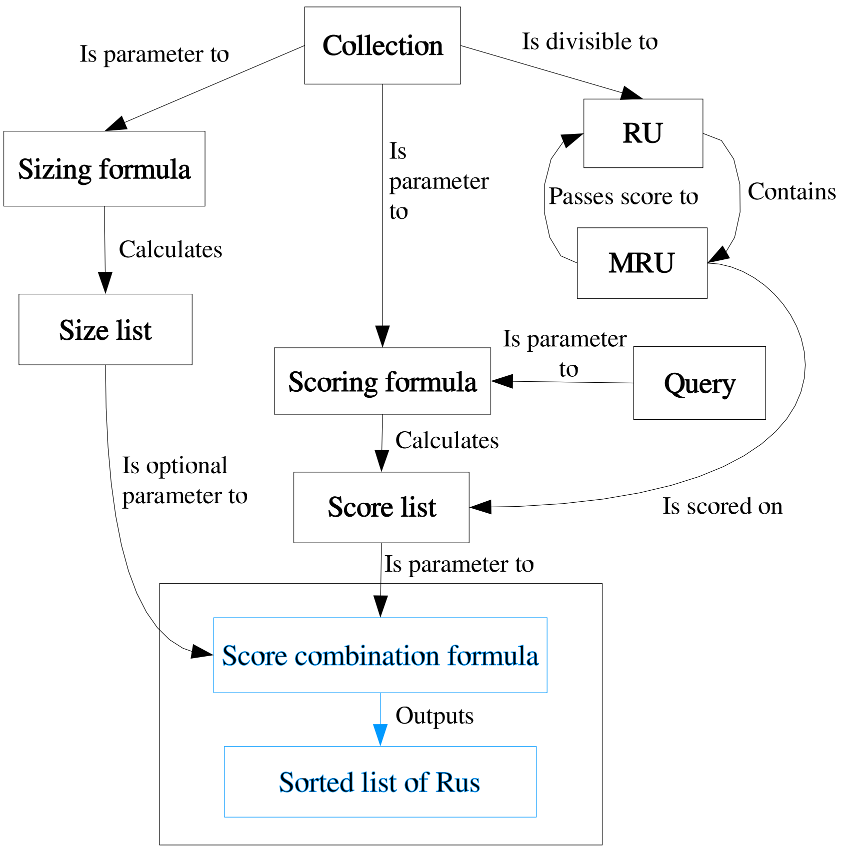


Figure 1: Conceptual relations of components both within and without the system

As the figure indicates, there is more than one way to count both the sizes of RUs and the scores of MRUs. For example, the amount of textual characters inside the document fragment is one indicator of size, and the number of words is another. Since a query may weight certain keywords negatively, a document fragment with such content may get a negative score; still, the formulae will work.

Let us consided the case where some document fragment contains more than one relevant MRU as its descendants. What is clearly needed is some way to determine whether that document fragment is more (or less) relevant than any (or all) of its component MRUs.

One such score combination method - know as *Upward Propagation* - simply sums the scores of all of the RU's descendant MRUs, and then scales the sum by the RU's size to the power of a given parametric factor:

$$score(a) = \frac{\sum score(m)}{size^p(a)} \tag{1}$$

where a is the ancestor RU, m are its immediate descendant RUs and p is the RSV upward propagation factor (see *Terminology*). Here the underlying assumptions are that relevance is proportional to size, and the amount of redundant material must be kept in check. As already mentioned, the size of a RU can be defined in several ways. Another thing worth noting is the fact that, as the name *Upward Propagation* suggests, the processing has to proceed bottom-up - starting from the MRU leaves and scoring other RUs only after all their immediate descendants are scored.

The scores can also be weighted according to RU branching, i.e. the number of its children:

$$R = D(n) \sum{}_{i=1 \ldots n} L_i \tag{2}$$

where R stands for relevance, n is the number of child RUs, $D(n)$ is 0.49 if n is 1 and 0.99 otherwise, and L_i is the child RU score.

Here the function of branching effects a simple step between two weights, to penalize those parent RUs that do not actually combine *scores* but rather inherit a single score; weighing down such a parent helps remove some of the overlapping content in the ranked list of answers. Some similar method that employed a continuous (maybe logarithmic) function of branching would likely produce satisfying results.

These size and branching approaches *could* even be combined, although such a formula has not been provided.

After the scores are calculated, the fragments are sorted before a listing of 1500 best answers is returned. Sorting is done by selecting the document fragment with highest remaining score, adding it to the list and removing its descendants from the selection. This is repeated until 1500 document fragments are listed in order of relevance. If at some point there are two or more document fragments sharing the highest score, the one with a shorter path has precedence. In the highly unlikely event that even the paths are of equal length, either of the document fragments is added because it is known not contain the other. Finally, if the selection is exhausted before 1500 document fragments are listed, the previously removed descendants come back into consideration.

Customers: is this more or less correct?

3 Requirements on the system

The requirements on ScoreCombiner are divided into functional, system and qualitative requirements. Listed requirements take the following form:

Identifier	R0
Name	Requirement name
Description	Requirement description.
Priority	Requirements have been prioritized in co-operation with the clients, to a scale of 1–4 where 1 is the highest and 4 the lowest priority
Functionality	Reference (i.e. function identifier) to functions of Ch. 4 that implement this requirement or existing Perl scripts (P)

3.1 Functional requirements

The functional requirements are presented in the order in which they are encountered on ScoreCombiner.

3.1.1 Input

Identifier	R1
Name	Input directory
Description	The directory from which ScoreCombiner looks for input files is changeable.
Priority	2
Functionality	

Identifier	R2
Name	User preferences
Description	User defines *separate* input directories for size lists, score lists, map files and formulae, as well as an output directory for the results. All of these are implemented *as system properties*.
Priority	2
Functionality	

Identifier	R3
Name	Default input directory
Description	Current input directory may be set as default. If none is set, ScoreCombiner's location acts as default input directory.
Priority	3
Functionality	-

Identifier	R4
Name	Score list
Description	Any of the score lists available in input directory may be selected.
Priority	1
Functionality	F6, Score list panel; F7, Selecting a score list

Identifier	R5
Name	Size list
Description	Any of the size lists available in input directory may be selected.
Priority	1
Functionality	F8, Size list panel; F9, Selecting a size list

Identifier	R6
Name	Score combination formula
Description	Any of the score combination formulae available in input directory may be selected.
Priority	1
Functionality	F11, Formula panel; F13, Selecting a formula

Identifier	R7
Name	Input history
Description	A "most-recently-used" list is kept of used score lists, size lists and score combination formulae; new entries are added to the top of the list, and old ones are simply moved there.
Priority	2
Functionality	F3, File history

Identifier	R8
Name	Validity of history
Description	Each entry in input history corresponds to a valid, existing input file.
Priority	3
Functionality	F3, File history

Identifier	R9
Name	Formula recognition
Description	Available formulae are automatically found from a directory that is exclusive to them. Upward Propagation is required for ScoreCombiner to launch.
Priority	1
Functionality	F12, Formula collection; F11, Formula panel

Identifier	R10
Name	List recognition
Description	Available score and size lists are automatically found from a directory that is exclusive to them.
Priority	2
Functionality	F6, Score list panel; F8, Size list panel

3.1.2 Formulae

Identifier	R11
Name	Upward Propagation
Description	The formula of Upward Propagation is implemented as it is described in this document (see Equation 1), as an extension of the formula superclass.
Priority	1
Functionality	F12, Formula collection

Identifier	R12
Name	Formula addition
Description	New formulae are added through simple extensions of the formula superclass that are compiled outside ScoreCombiner.
Priority	1
Functionality	F17, Formula - *Group: new function?*

Identifier	R13
Name	Accessing scores and sizes
Description	Formula superclass provides an interface for a formula to access document fragments' score and size when required. Indirect access through score and size lists is sufficient.
Priority	1
Functionality	F17, Formula

Identifier	R14
Name	Accessing names and branching
Description	Formula superclass provides an interface for a formula to access document fragments' element name and branching (number of children) when required. Indirect access through score and size lists is sufficient.
Priority	2
Functionality	F17, Formula

3.1.3 Parameters

Identifier	R15
Name	Score combination parameter setting
Description	Parameters can be given to score combination formulae that support them.
Priority	1
Functionality	F15, Setting the parameter values; F14, Parameter panel

Identifier	R16
Name	Information on formula parameters
Description	Formulae provide the needed information on how many parameters they need, what type they are and what other restrictions concern their value range.
Priority	3
Functionality	F17, Formula - *Group: new function?*

Identifier	R17
Name	Defaulting to valid formula parameters
Description	Formulae that need parameters provide valid default values for them.
Priority	2
Functionality	F16, Setting the default parameter values

Identifier	R18
Name	Remembering formula parameters
Description	A record is kept of preferred user parameters for each used formula.
Priority	2
Functionality	-

Identifier	R19
Name	GUI for formula parameter entry
Description	ScoreCombiner generates a GUI for entering parameters, based on information provided by the formula.
Priority	1
Functionality	F14, Parameter panel

3.1.4 Processing

Identifier	R20
Name	Run queue
Description	A new run using the given lists, formula and parameters is added to a queue of runs. The runs are processed in order of entry.
Priority	2
Functionality	F22, Creating a score combination process

Identifier	R21
Name	Run status
Description	The name, input files and parameters used in the run are displayed in a list. The same list contains both waiting *and* already completed runs displayed in different colours. If an error was encountered during a run, its status is displayed in red.
Priority	2
Functionality	F24, Information panel

Identifier	R22
Name	Status log
Description	Completed runs are appended into a separate text file. In addition to name, input files and used parameters, information is provided on possible errors.
Priority	2
Functionality	F24, Information panel

Identifier	R23
Name	Scoring the collection
Description	Entire collection is scored in bottom-up fashion - applying the score combination formula on each document fragment and receiving comparable scores for them.
Priority	1
Functionality	F23, Running the score combination process

Identifier	R24
Name	Selection of scores to output
Description	A ranked list of 1500 highest scores is created. Repeatedly, the document fragment with highest remaining score is added to the list and removed from selection along with its descendants, until the list is complete. In case the highest score is not unique, the document that also has shortest path is added.
Priority	1
Functionality	-

3.1.5 Output

Identifier	R25
Name	Output directory
Description	The output files are stored into user-defined directories.
Priority	1
Functionality	-

Identifier	R26
Name	Default output directory
Description	Current output directory may be set as default. If none is set, ScoreCombiner's location acts as default output directory.
Priority	2
Functionality	-

Identifier	R27
Name	Score information
Description	The year of query is determined from suffix of used score list and embedded into the output document.
Priority	1
Functionality	-

Identifier	R28
Name	Query answer
Description	The ranked list of 1500 most relevant document fragments is embedded into the output document.
Priority	1
Functionality	-

3.2 System requirements

System requirements define the performance-related bounding values like response times and file sizes, as well as some of the restrictions and limitations of ScoreCombiner.

3.2.1 Files

Identifier	R29
Name	Map input file
Description	The map file represents the collection. The map file is recognized from file suffix .map.
Priority	1
Functionality	-

Identifier	R30
Name	Map input file format
Description	Each line in the map file has two xml elements. First element tells the path of file in collection. Second tells the path of an xml element in that file. Lower level elements are implicitly covered in paths of upper level elements.
Priority	1
Functionality	-

Example 1
The map file
<file>tp/2002/i0001</file><path>/article[1]/fm[1]</path>
<file>tp/2002/i0001</file><path>/article[1]/bdy[1]/sec[1]</path>
<file>tp/2002/i0001</file><path>/article[1]/bdy[1]/sec[2]</path>
. . .

Identifier	R31
Name	Score list input file
Description	The score list file contains pre-calculated scores for minimal retrieval units for each query. The score file has a file suffix like .Aggreg.AllScores.2004.
Priority	1
Functionality	-

Identifier	R32
Name	Score list input file format
Description	The score file starts from submission a tag and ends to the matching ending tag. The scores are wrapped inside topic elements. The Topic elements represent different queries and are distinguished by integer attributes. Inside topic elements there, is a list of decimal and integer number pairs separated by a tab character. The decimal number is the score of element. The integer is a so called fragment index number. It binds the scores to the lines of map file.
Priority	1
Functionality	-

Example 2

The score list file

<submission>
<topic 162>
0.740836526117954 89920
0.924652083073702 89923
0.599337302824778 89925
. . .
</topic>
<topic 163>
. . .
</topic>
</submission>

Identifier	R33
Name	Size list input file
Description	The size file contains pre-calculated size for each element in collection. The Size file has file suffix .AllSizes.
Priority	1
Functionality	-

Identifier	R34
Name	Size list input file format
Description	The Format is much like the the format of the map file. At the end of the each line, the size information is presented. It is an integer, separated by one tab character. Size information is calculated for all elements in collection, not just for minimal retrieval units.
Priority	1
Functionality	-

Example 3
The size list file
<file>tp/2002/i0001</file><path>/article[1]/fm[1]</path> 2120
<file>tp/2002/i0001</file><path>/article[1]</path> 53431
. . .

Identifier	R35
Name	Score combination formula input file
Description	All score combination formulae are Java classes - instances of the formula superclass that implement possible parameter handling and GUI for their entry as well as the actual calculation. Their return values are within specified ranges.
Priority	1
Functionality	-

Identifier	R36
Name	Output file
Description	Output file is produced as a result of the calculation process. The file contains 1500 of the most relevant elements of evaluated documents for each query.
Priority	1
Functionality	-

Identifier	R37
Name	Output file format
Description	Output file is an XML file. See result.dtd for details.
Priority	1
Functionality	-

Example 4
The Output file
<topic topic-id="91">
<result>
<file>ic/2001/w6070</file>
<path>/article[1]/bm[1]</path>
<rsv>0.944420711298685</rsv>
</result>
. . .

3.2.2 Response times

Identifier	R38
Name	Launching ScoreCombiner
Description	After The ScoreCombiner is launched, it is operational in 10 seconds.
Priority	1
Functionality	-

Identifier	R39
Name	User interaction
Description	Changing input and possible parameters takes effect in less than one second.
Priority	1
Functionality	-

Identifier	R40
Name	Duration of processing
Description	Processing times are comparable to those of existing Perl scripts and not significantly worse.
Priority	1
Functionality	P

3.3 Qualitative requirements

Unlike other types of requirement, qualitative one do not have a priority (that is, their priority is fixed at the highest value 0) nor have they any references to specific functions; rather, their effect is felt on the entire system.

3.3.1 Useability

Identifier	R41
Name	No user levels
Description	ScoreCombiner has only one user level. All users have similar access to the provided services.
Priority	0
Functionality	-

Identifier	R42
Name	GUI completeness
Description	The implementation of ScoreCombiner does not limit the intended use or specified value range of the formula. The formula itself *may* impose such limitations.
Priority	0
Functionality	-

3.3.2 Maintenance

Identifier	R43
Name	Understandability
Description	The process, documentation and coding on this software development project are in English.
Priority	0
Functionality	-

Identifier	R44
Name	Well-documented code
Description	All delivered code is well-documented. Sections related to the addition of new formulae and produced output are clearly pointed out in the documentation.
Priority	0
Functionality	-

Identifier	R45
Name	Changeability
Description	Changing a single, *well pointed-out* constant value in the source code and recompiling changes the number of document fragments included in the output file.
Priority	0
Functionality	-

Identifier	R46
Name	Expandability
Description	New formulae are added by extending the formula superclass and compiling it. The possibility of this is tested before implementation.
Priority	0
Functionality	-

3.3.3 Other

Identifier	R47
Name	Homogeneity
Description	Code will be produced consistently using version 1.4 of Java with no deprecation.
Priority	0
Functionality	-

Identifier	R48
Name	Correctness
Description	ScoreCombiner is sufficiently tested on actual material using accurate methods.
Priority	0
Functionality	-

3.4 Requirements listing

The following table contains the requirements in prioritized order.

0 R41, No user levels	0 R42, GUI completeness
0 R43, Understandability	0 R44, Well-documented code
0 R45, Changeability	1 R4, Score list
1 R5, Size list	1 R6, Score combination formula
1 R9, Formula recognition	1 R11, Upward Propagation
1 R12, Formula addition	1 R13, Accessing scores and sizes
1 R15, Score combination parameter setting	1 R19, GUI for formula parameter entry
1 R23, Scoring the collection	1 R24, Selection of scores to output
1 R25, Output directory	1 R27, Score information
1 R28, Query answer	1 R29, Map input file
1 R30, Map input file format	1 R31, Score list input file
1 R32, Score list input file format	1 R33, Size list input file
1 R34, Size list input file format	1 R35, Score combination formula input file
1 R36, Output file	1 R37, Output file format
1 R38, Launching ScoreCombiner	1 R39, User interaction
1 R40, Duration of processing	2 R1, Input directory
2 R2, User preferences	2 R7, Input history
2 R10, List recognition	2 R14, Accessing names and branching
2 R17, Defaulting to valid formula parameters	2 R18, Remembering formula parameters
2 R20, Run queue	2 R21, Run status
2 R22, Status log	2 R26, Default output directory
3 R3, Default input directory	3 R8, Validity of history
3 R16, Information on formula parameters	

4 Functionality

System functions are based on the requirements stated in chapter 3.

Functions are arranged under main, history, score list, size list, map file, formula, output and score combination categories. Stated functions take the following form:

Identifier	F0
Name	Name of the function
Description	Description of the function.
Initial state requirement	Initial state requirements of the function
Input	Input required by the function
Output	Output produced by the function
System state change	System state changes caused by the function
Priority	Functions have been prioritized in co-operation with the clients, to a scale of 1–4 where 1 is the highest and 4 the lowest priority
Implemented requirements	Reference (i.e. function identifier) to requirements from Ch. 3)

4.1 Relations between functions

A picture will be added here.

4.2 Main window

Identifier	F1
Name	Main window
Description	Creates system's main window. See appendix 1 for more detailed information.
Initial state requirement	ScoreCombiner has been started.
Input	-
Output	Score list panel, size list panel, formula panel, parameter panel, output panel, score combination button and information panel.
System state change	-
Priority	1
Implemented requirements	see panels' implemented requirements

4.3 Browsing

Identifier	F2
Name	Browsing
Description	A score list file or a size list file is selected by browsing.
Initial state requirement	-
Input	Default input directory.
Output	The selected file appears in the top of the file history.
System state change	A new file has been selected. F5, History addition is done.
Priority	2
Implemented requirements	R1, Input directory; R4, Score list; R5, Size list

4.4 Functions related to the file history

Identifier	F3
Name	File history
Description	Creates a collection of most recently used files where one file can be selected at a time. Appears in score list panel and in size list panel.
Initial state requirement	-
Input	A list of paths and file names.
Output	A combo box containing not more than 10 most recently used files. The name of the file used last time appears as selected file.
System state change	The non-existing files are removed from the list. A file has been selected if the history is not empty.
Priority	2
Implemented requirements	R7, Input history; R8, Validity of history

Identifier	F4
Name	History selection
Description	Selects a file from the file history.
Initial state requirement	File history has been created and at least two files appear in the history.
Input	File
Output	File's name appears as selected file.
System state change	Selection has changed. The selected file is moved to the top of the file history.
Priority	2
Implemented requirements	

Identifier	F5
Name	History addition
Description	A file is added to the file history.
Initial state requirement	A file has been chosen outside the file history.
Input	A file and knowledge of history type (score/size).
Output	The added file appears in the top of the file history or an error message containing knowledge of file not being the needed type of file and not being added to the history, appears in the same panel where the history does.
System state change	If the file is valid according to the type of the history, it is added to the top of the file history and the last file in the file history is removed if the history contains more than 10 files.
Priority	2
Implemented requirements	R8, Validity of history

4.5 Functions related to the score list

Identifier	F6
Name	Score list panel
Description	Creates a panel where a score list for score combination can be chosen by browsing or by selecting one from the score list history.
Initial state requirement	-
Input	-
Output	Score list history and a browsing button.
System state change	-
Priority	1
Implemented requirements	R1, Input directory; R4, Score list

Identifier	F7
Name	Selecting a score list
Description	Selects a score list for score combination.
Initial state requirement	-
Input	Score list file.
Output	The file appears as selected score list.
System state change	F2, Browsing or F4, History selection is done. Score list for score combination formula has been changed. F10, Finding the map file is done.
Priority	1
Implemented requirements	R4, Score list

4.6 Functions related to the size list

Identifier	F8
Name	Size list panel
Description	Creates a panel where the size list for score combination can be chosen by browsing or by selecting one from the size list history.
Initial state requirement	-
Input	-
Output	Size list history and a browsing button.
System state change	-
Priority	1
Implemented requirements	R1, Input directory; R5, Size list

Identifier	F9
Name	Selecting a size list
Description	Selects a size list for score combination.
Initial state requirement	-
Input	Size list file.
Output	The file appears as selected size list.
System state change	F2, Browsing or F4, History selection is done. Size list for score combination formula has been chosen.
Priority	1
Implemented requirements	R5, Size list

4.7 Functions related to the map file

Identifier	F10
Name	Finding a map file
Description	Finds the map file related to the selected score list.
Initial state requirement	Score list has been selected.
Input	Score list. Input directory for map files.
Output	An error message in score list panel containing information on missing map file, if no map file for score list is found.
System state change	Map file for score combination has been chosen if the file exists.
Priority	1
Implemented requirements	

4.8 Functions related to the score combination formula

Identifier	F11
Name	Formula panel
Description	Creates a panel where a formula for score combination can be chosen from by selecting one from formula collection.
Initial state requirement	-
Input	-
Output	Formula collection. Parameter panel if the selected formula needs parameters.
System state change	Parameter panel is created if the formula needs parameters.
Priority	1
Implemented requirements	R6, Score combination formula

Identifier	F12
Name	Formula collection
Description	Creates a collection of formulas. All formulas not corresponding to R35, Score combination formula input file are discluded from the collection.
Initial state requirement	-
Input	Input directory for formulas and knowledge of the most recently used formula.
Output	A combo box containing all the formulas. The most recently used formula appears as the default selection if it exists. If not, the collection's first formula appears as one.
System state change	If the most recently used formula desn't exist, the first formula in the collection is set as one. If the collection is not empty, a formula has been chosen.
Priority	2
Implemented requirements	R6, Score combination formula; R9, Formula recognition

Identifier	F13
Name	Selecting a formula
Description	Selects a formula for score combination.
Initial state requirement	-
Input	Formula
Output	Parameter panel if the chosen formula needs parameters. The input formula appears as the selected formula.
System state change	Parameter panel is created into the formula panel if the chosen formula needs parameters. If the parameter panel exists and the formula doesn't need any parameters, it's removed from the formula panel.
Priority	1
Implemented requirements	R6, Score combination formula

Identifier	F14
Name	Parameter panel
Description	Creates a panel where one or more parameter values for the score combination formula can be set based on R16, Information on formula parameters.
Initial state requirement	Formula needs parameters.
Input	Parameter types, descriptions and default values.
Output	Input fields for the parameters with default values (if any has been defined) and information about them and a save button for setting new default parameter values for the formula.
System state change	-
Priority	1
Implemented requirements	R15, Score combination parameter setting; R16, Information on formula parameters; R17, Defaulting to valid formula parameters; R19, GUI for formula parameter entry

Identifier	F15
Name	Setting the parameter values
Description	Sets one or more parameter values for the score combination formula.
Initial state requirement	Formula needs parameters.
Input	Parameter values.
Output	Parameter values appear in the parameter fields.
System state change	New parameters for the score combination formula have been chosen.
Priority	1
Implemented requirements	R15, Score combination parameter setting

Identifier	F16
Name	Setting the default parameter values
Description	Sets the new default values for formula's parameters by providing them to the formula.
Initial state requirement	Formula needs parameters.
Input	Values.
Output	An error message containing information on wrong parameter types or/and value being out of the range and the current default values provided by the formula to the parameter panel if the new values are not valid.
System state change	New default parameter values for the formula are set if the defined values are valid.
Priority	3
Implemented requirements	

Identifier	F17
Name	Formula
Description	Counts the scores for all non-minimal RUs.
Initial state requirement	Formula has been invoked by a score combination process.
Input	Parameters if formula needs any.
Output	A list containing all non-minimal RUs and their scores.
System state change	the output list is returned to the invoker.
Priority	1
Implemented requirements	R13, Accessing scores and sizes; R14, Accessing names and branching

4.9 Functions related to the resulting score list

Identifier	F18
Name	Output panel
Description	Creates a panel where a path and a name for the output file can be given. Also a description for the score combining can be given.
Initial state requirement	-
Input	Knowledge of the most recently used path.
Output	A text field for the output file's path and name, a text field for the description. The most recently used path appears in the text field as default path. For the first time user's home directory is the default directory.
System state change	-
Priority	2
Implemented requirements	R25, Output directory

Identifier	F19
Name	Defining the result file
Description	Defines the location and/or the name of the output file.
Initial state requirement	-
Input	File name and/or path.
Output	Changed location and/or a new name appear in the output panel's text field.
System state change	The name and/or the path for the output file has been changed.
Priority	2
Implemented requirements	R25, Output directory

Identifier	F20
Name	Adding a description
Description	Adds a description for the score combination.
Initial state requirement	-
Input	Description.
Output	Description appears in the output panel's description field.
System state change	Description for the score combination has been set.
Priority	2
Implemented requirements	

4.10 Functions related to the score combination

Identifier	F21
Name	Checking the values
Description	Checks the validity of the parameters (if formula needs any) and the output file's path and name (if one has been defined).
Initial state requirement	Button for score combination has been clicked and a score list, a size list, a map file and a formula have been chosen.
Input	A path, parameters if formula needs any and a name if it has been defined.
Output	Default values and an error messages for false ones. The error message for non valid parameter values contains information on false parameter types or the values being out of range. The error message for non valid output file definition contains information on incorrect path or incorrect name.
System state change	If the parameter values are not valid, they are removed (if no default values have been defined) or replaced with the default values by consulting the formula. The non valid path is replaced with the default output path. If the values are ok, F22, Creating a score combination process is done.
Priority	1
Implemented requirements	R17, Defaulting to valid formula parameters

Identifier	F22
Name	Creating a score combination process
Description	Creates a scrore combination process.
Initial state requirement	F21, Checking the values has been done.
Input	Score list, size list, map file, formula, (parameters), output file's path (and name and description for process).
Output	The output file name and the phase of the combination into the information panel's process list.
System state change	The process is started and its status is set to *Running* if no other process is running. Else the process is added to a queue where it waits for its turn and its status is is set to *Waiting*.
Priority	1
Implemented requirements	R20, Run queue

Identifier	F23
Name	Running the score combination process
Description	Runs the score combination formula. Creates the result score list and writes it and the description to the output file according to R37, Output file format. Knowledge of used files and formula are added to the description. Writes a log file according to R22, Status log. If no name was defined for the output file, one is created according to ??.
Initial state requirement	The process has been started or popped out of the queue.
Input	Process.
Output	Output file, information on process to the information panel and a log file. Process' error status is displayed in red.
System state change	Process' status is changed to *Finished* when the process has been completed.
Priority	1
Implemented requirements	R23, Scoring the collection; R24, Selection of scores to output; R27, Score information; R28, Query answer; R36, Output file; R37, Output file format; R25, Output directory; R20, Run queue

Identifier	F24
Name	Information panel
Description	Creates a list where information on the score combination processes according to R21, Run status can be added to.
Initial state requirement	-
Input	-
Output	An empty list of score combination processes.
System state change	-
Priority	3
Implemented requirements	R21, Run status

4.11 List of functions

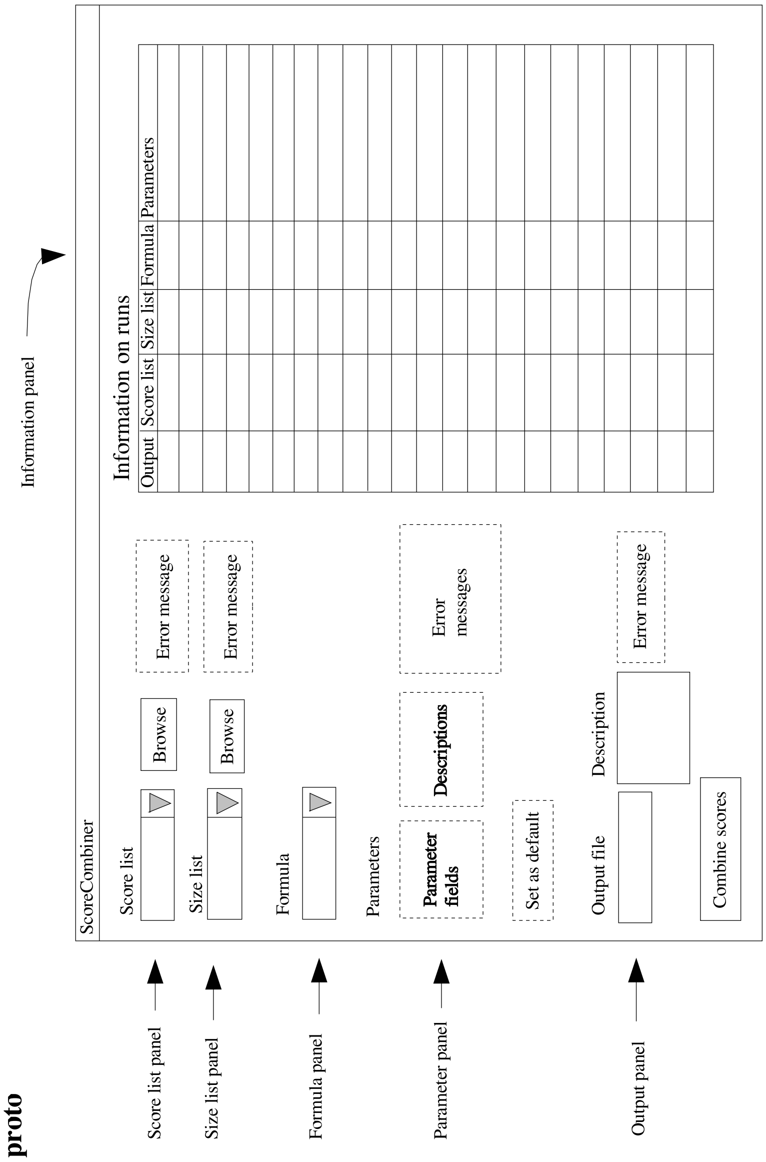
Following table shows the functions in prioritized order.

1 F1, Main window	1 F6, Score list panel
1 F7, Selecting a score list	1 F8, Size list panel
1 F9, Selecting a size list	1 F10, Finding the map file
1 F11, Formula panel	1 F13, Selecting a formula

1 F14, Parameter panel	1 F15, Setting the parameter values
1 F17, Formula	1 F21, Checking the values
1 F22, Creating a score combination process	1 F23, Running the score combination process
2 F2, Browsing	2 F3, File history
2 F4, History selection	2 F5, History addition
2 F12, Formula collection	2 F18, Output panel
2 F19, Defining the result file	2 F20, Adding a description
3 F16, Setting the default parameter values	3 F24, Information panel

References

Appendix 1. UI proto



UI proto

Appendix 2. result.dtd

```
<!ELEMENT inex-submission (description?, topic+)>
<!ATTLIST inex-submission
     participant-id  CDATA         #REQUIRED
     run-id          CDATA         #REQUIRED
     task   ( CO | VCAS )          #REQUIRED
     query  (automatic | manual)   #REQUIRED
>

<!ELEMENT description (#PCDATA)>

<!ELEMENT topic (result*)>
<!ATTLIST topic
     topic-id        CDATA  #REQUIRED
>

<!ELEMENT result (file, path, rank?, rsv?)>
<!ELEMENT file (#PCDATA)>
<!ELEMENT path (#PCDATA)>
<!ELEMENT rank (#PCDATA)>
<!ELEMENT rsv  (#PCDATA)>
```