Requirements document

DaCoPAn2

Helsinki, 24th March 2005

Software Engineering Project

UNIVERSITY OF HELSINKI
Department of Computer Science

Course

581260-4 Software Engineering Project (6 cr)

Project Group

Mikko Airaksinen
Tomi Korkki
Pauli Miettinen
Timo Tuominen
Mikko Väänänen

Customer

Markku Kojo

Project Masters

Juha Taina (Supervisor)
Marianne Korpela (Instructor)

Homepage

`http://www.cs.helsinki.fi/group/dacopan2`

Change Log

Version	Date	Modifications
0.50	23.02.2005	Rough outline
0.90	07.03.2005	First complete version
1.00	09.03.2005	Finished version sent to customer
1.10	24.03.2005	Final and corrected version

Contents

1 Introduction

DaCoPAn2 is a follow-up project of the Animator subsystem produced by the original DaCoPAn software engineering project. The DaCoPAn Animator is a software which illustrates standard network communication protocols using different animation types.

1.1 Purpose of this document

This requirements document is a specification of the requirements for the DaCoPAn2 software engineering project. It serves as a contract between the project team and the customer, and spells out the general guidelines for the future design and implementation phases of the project.

1.2 Scope of the Development Project

Since all the requirements for DaCoPAn2 are based on already existing software, this document focuses on additional features and improvements requested by the customer; additionally, it specifies the constraints that the original software imposes on new features. For further specifications of the Animator requirements not included in this document refer to the Requirements Specification of the DaCoPAn project [2].

The major requirements for DaCoPAn2 are: implementing a new Time Sequence Chart (TSC) animation type as an additional visualization, making major improvements to the existing Message Sequence Chart (MSC) animation type and enhancing other user interface features, including the handling of program settings.

1.3 Overview of the Document

The document is organized into the following sections:

The document begins with, section 2, a brief description and decomposition of the existing DaCoPAn software version. The constraints that the current version imposes on extensions are also discussed. In subsection 2.2, high level requirements of the DaCoPAn2 project are also introduced.

Following this, in section 3, the actual user requirements and system functionalities are specified. For the two main components involved in the project, the MSC and TSC views, an overview of the animation types and their user interfaces are included.

It is recommended that the reader start by reading the overview of each section.

2 The DaCoPAn software

In this section, the existing version of the DaCoPAn software is described. The two subsystems, the Analyzer and the Animator, and their common interface are introduced. Since DaCoPAn2 focuses entirely on the Animator subsystem, in section 2.2, a more detailed decomposition of its structure is presented along with high level requirements of this project.

2.1 Overview

The DaCoPAn software was created to analyze log information captured from actual network communication between two hosts, and to visualize this data as different animation types. Its intended main user groups are educators, researchers and students. However, the DaCoPAn2 project will primarily focus on improving educational aspects of the software.

Below, in Figure 1 from the DaCoPAn project's Requirements Specification [2], is a general view of the DaCoPAn software.

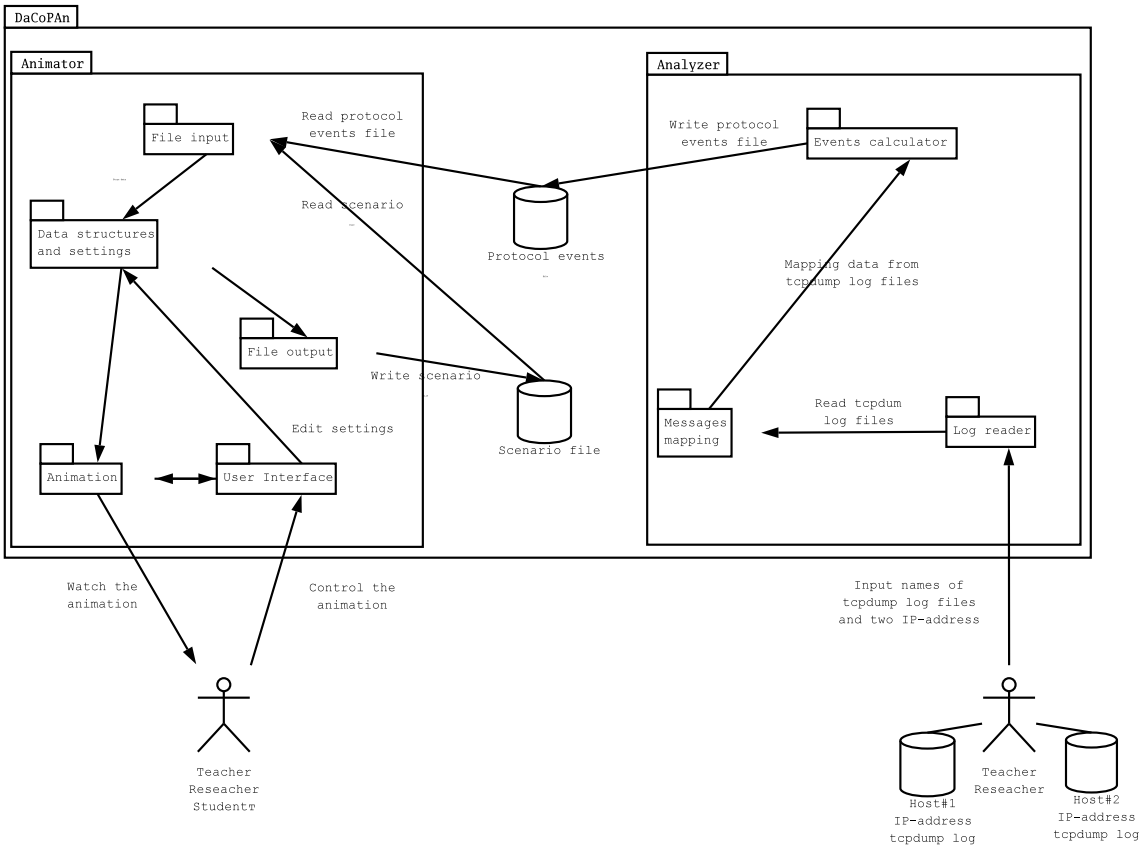

Figure 1: High level model of the DaCoPAn system

The DaCoPAn software consists of two subsystems:

- The **Analyzer** combines the data from two *tcpdump* log files representing each host's point of view of the network connection. The Analyzer stores the collected data into an intermediate Protocol Event File (PEF), the format specification of which acts as an interface between the Analyzer and Animator.
- The **Animator** reads the data produced by the Analyzer into its internal data structures, and presents this data to the user as visual representations that are animated over time and displayed in the program window.

2.2 The Animator subsystem

In this section, the existing Animator subsystem is described in more detail. The high level requirements of the current project are also specified.

Below, in Figure 2, is a top level data flow diagram of the Animator. The Animator subsystem is represented as a central process, and the external entities to the process are represented by rectangular boxes. The arrowed lines represent data flows.

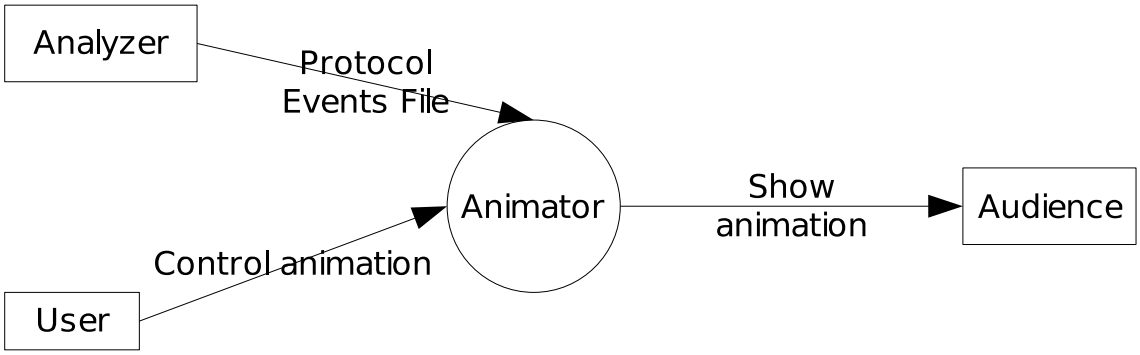

Figure 2: Level 0 data flow diagram of the Animator

On the next page, Figure 3, is a more fine grained presentation of the Animator process of the top level diagram. The notation includes some data stores, represented as open-ended rectangles.

The Animator implementation can be organized into the following logical parts, and each part, as it relates to our project requirements, is briefly discussed here.

2.2.1 Program input and data representation

The Animator takes PEF files as its primary input. The internal data model classes, which model the network communication, are populated from the data found in the PEF. The format of the PEF file has been specified in [1]. DaCoPAn2 will follow this specification. Some user requirements have to do with data not currently supported

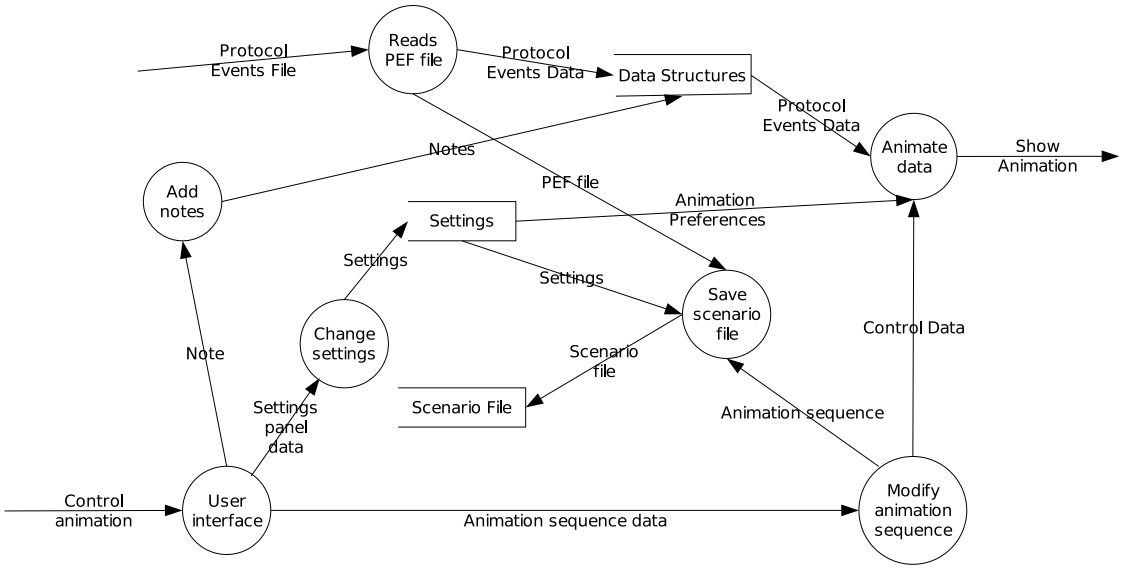

Figure 3: Level 1 data flow diagram of the Animator

by the Analyzer. In these cases the existing extension support of the PEF file specification will be used and new variables introduced.

2.2.2 User interface

The main restrictions imposed on the project by the existing software have to do with the way the user interface and the different animation types are implemented. The user interface is implemented using Java Swing, and the new elements added will have to conform to the existing user interface class hierarchy. Also, existing conventions in the user interface have to be used in any new components unless explicit user requirements state otherwise.

2.2.3 Animation types

The animation types are different visualizations of the internal data model. The largest modifications required to the software concern the Message Sequence Chart (MSC) animation type. A more detailed description of this animation type is found in section 3.2.2.

The new animation type, the Time Sequence Chart (TSC), is introduced in section 3.2.1 and will have to conform to the existing class hierarchy of the Java code; however, it will be designed and implemented as a completely new component.

Other existing animation types include the Encapsulation (ENC) view and the Unit Flow Orchestration (UFO) view. Improvements to the Encapsulation view have been ruled out at present, but some functionalities concerning the UFO view are to be implemented.

2.2.4 Settings

The Animator has a separate settings panel in the user interface. The panel handles both global settings and the user preferences for displayed unit information in the MSC panel. New tabs will be added to the panel, specifically for the TSC animation type, and old tabs will be refined.

The existing software has no properly functioning default settings and no way to save global settings. The way global and animation type-specific settings are handled will be improved, so that the reiterating of already chosen user settings will be eliminated.

2.2.5 Scenarios

The internal handling of user settings is linked to scenarios, which are animation sequences where the user has added notes and break points. The user settings are presently saved only in the scenario files.

User added notes will be an essential part of the TSC animation type, and support for automatically triggered user editable notices will be added in the existing scenario framework.

3 User requirements and system functions

In this section, the user requirements and system functions are described. The section is further divided according to the program categories the requirements belong to. In addition, some general functional and non-functional requirements are specified.

3.1 Explanation of document conventions

Every listed requirement uses the following format:

Requirement The code for the requirement.
Name The name of the requirement.
Priority The priority of the requirement.

A textual description of the requirement.

Function The code for the function.

A textual description of the function.

Note: Reason why not implemented.

Above, the requirement code is an individual identification of the requirement. For example **E1.2**, where E specifies the program category in question and this is followed by the requirement number. The categories are noted below. The name of the requirement is simply a brief descriptive caption. The priority indicates the importance of the requirement. The priorities are listed on the following page. Below the requirement description, functions corresponding to the requirement are described along with their codes. If the requirement has been chosen not to be fulfilled due to low priority, or has no detailed functionality other than repeating the requirement text, the function descriptions are omitted. The note is a brief explanation for why the requirement was chosen not to be implemented. This only applies if the listed priority is P4.

The program categories used in the codes are:

A Time Sequence Chart (TSC)

B Message Sequence Chart (MSC)

C Encapsulation view (ENC)

D Unit Flow Orchestration view (UFO)

E Settings

F General

There are four different priorities classifying the importance of each described requirement. The same classification applies to system functionalities. These are:

P1 Must be implemented: These are requirements and functionalities that have been assessed as being critical, in order to have a successful completion of the project. They are defined by the customer.

P2 Preferrably implemented: These requirements and functionalities are to be implemented if feasible; this assumes that the time constraints and existing code allow for such. These are suggestions made by the customer.

P3 Optional: These requirements and functionalities are to, possibly, be implemented if time allows. These are generally improvements the group has come up with on its own, or may be suggestions by the customer.

P4 Not to be implemented: These are requirements and functionalities that have been determined to be non-critical, or not feasible to implement due to time constraints and/or issues with code. Possibly left to be implemented by later continuing DaCoPAn projects. The reason for not implementing the feature will also be stated.

3.2 Functional requirements

In this section, the functional requirements and the corresponding functionalities are specified. The section is subdivided according to the program categories the requirements belong to.

3.2.1 Time Sequence Chart

The Time Sequence Chart (TSC) shows detailed information about the status of transport layer communication between two hosts. Its main function is to illustrate TCP layer communication.

In Figure 4, there is an outline of the Time Sequence Chart layout. The TSC view is divided into five components. These are: the toolbar, the drawing area, the unit info panel, the tabbed notes and legend panel, and the notice bar.

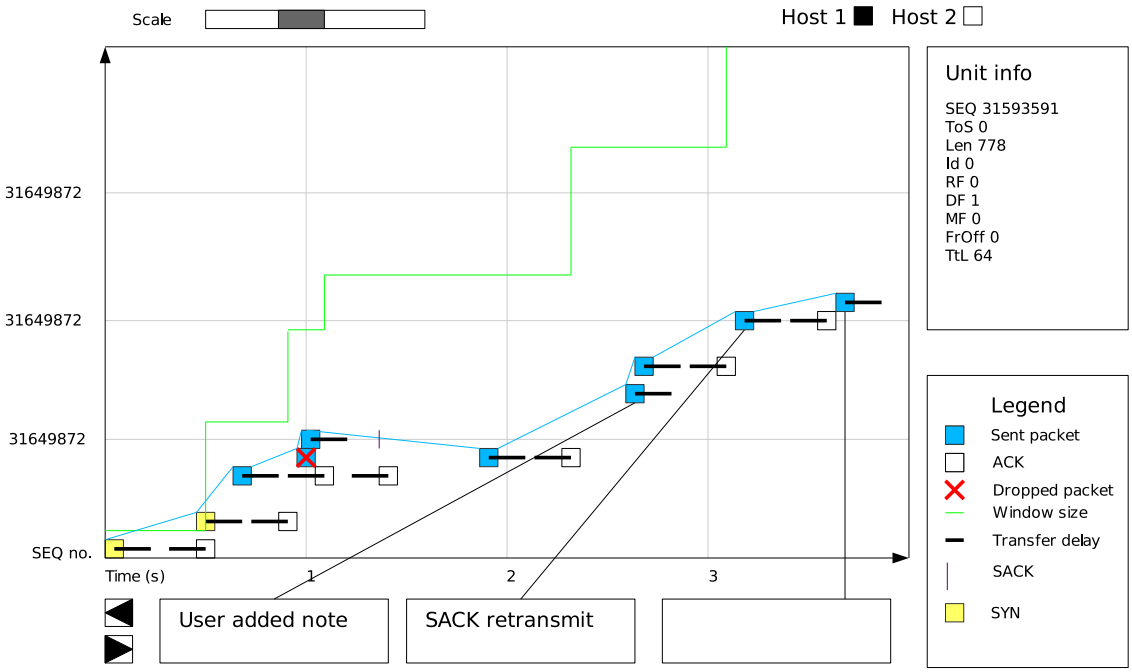

Figure 4: User interface of the TSC animation type

The horizontal dimension in the figure is the time axis, with the positive direction being from left to right. The vertical dimension is the sequence number of the packet, which increases from bottom to top.

TCP packets sent by hosts are symbolized by filled in squares. ACK packets are shown as empty squares, and dropped packets are marked by crossing over the packet in question. The TCP receiver advertised window size can be seen as a green line. The transfer delay is the dark horizontal line connected to the packet.

Requirement A1
Name Display of unit data
Priority P1

Unit variables are displayed in the unit info panel. The user can select the unit to be displayed.

Function A1.1
The unit data is shown by clicking on it when animation is paused. Data is displayed in a separate unit info panel. The user can choose which variables are shown using TSC-specific settings.

Requirement A2
Name Extending PEF files
Priority P1

Support must be added for TCP retransmission time-out and SACK block information.

Function A2.1
New data is added into PEF files by hand, since this project does not extend the Analyzator program.

Requirement A3
Name Notices
Priority P1

Notices contain brief free format textual information about a certain unit. Notices are attached to units and this relation must be seen from the TSC view. Notices are generated automatically from certain events which the user can choose from the settings.

Function A3.1
The animation stops for a moment when a new notice is displayed. The delay amount is determined from user-configured animation speed.

Function A3.2
Notices are attached to units using plain lines from the notice area to the graphical element.

Function A3.3
The notice text is duplicated at the top of the notes panel, from where they can be edited.

Requirement A4
Name Animation
Priority P1

The drawing area is animated in time sequential order. The screen follows the currently active unit.

Function A4.1
The drawing sequence for a unit is 1) unit symbol, 2) unit delay line. For the ACK packets, the sequence is 1) ACK delay line, 2) ACK symbol.

All the MSC buttons have the same logic in the TSC view. For example, you can step backwards and forwards in animation like in the MSC.

Animation speed is controlled by a "events per second" slider in the program settings. The animation is "user friendly" and uses delays between events, with active events being highlighted.

Requirement A5
Name Scaling
Priority P1

The user can configure the scaling to be used for the x-axis. The y-axis scale is adjusted automatically. The y-axis scale does not change during animation.

Function A5.1
Time/SEQNO axis labels scale automatically to a user friendly resolution preventing the numbers being too long. The time axis uses the following resolutions: microseconds (μs), milliseconds (ms) and seconds (s). The SEQNO axis uses an exponent scale if needed.

Function A5.2
The time axis scale varies from "minimum" to 100%. For example, "minimum" range means 10 events. The sequence number scale is determined using the sequence numbers from the first and last unit on the screen. Only one slider is used to configure scaling. The y-axis scale cannot change during the middle of animation.

Requirement A6
Name Relative SEQNO
Priority P2

The user can select if the sequence numbers start from 0 or follows the plain data sequence numbering.

Function A6.1
TSC-specific settings provide the option to set absolute SEQNO on/off. The pro-

gram must also be able to handle the sequence number wrapping over its 32-bit representation when in absolute mode.

Requirement A7
Name Colors and symbols
Priority P1

The user may choose the colors to be used in drawing area elements.

Function A7.1
A filled rectangle represents a unit. An outlined rectangle represents an ACK.

Symbols have a predefined minimum size, so that they are readable at all times using different scales. The user can choose the colors used for the graphical elements in the actual drawing area.

Requirement A8
Name Legend table for graphical elements
Priority P1

The display must contain a legend table.

Function A8.1
The Notes and Legend displays share the same space using tabbed panels.

Requirement A9
Name Choose displayed elements
Priority P1

TSC-specific settings provide the option of enabling or disabling the displayed graphical elements in the chart. The elements are:

1. **Units** The units are small filled-in rectangles. ACK units are empty rectangles.
2. **Connecting lines of units** Consecutive units are connected together by a line.
3. **Unit size** The units size is represented by the height of the rectangle, so that its size in bytes can be read from the vertical axis.
4. **Unit transfer delay line** The transfer delay of each packet is visualized by a horizontal line, which is drawn from the middle of the corresponding packet to the point in time where it has been fully transmitted. For ACK packets, the line starts when the ACK is sent from the remote host and ends in the middle of the ACK unit rectangle which depicts the receival of the ACK.
5. **SACK** The SACK blocks are drawn as vertical lines directly above the corresponding ACK packet. The lines' vertical coordinates represent the sequence numbers of the SACK blocks.
6. **SYN** The packets with SYN flags are highlighted using a different color chosen by the user from the settings.
7. **Advertised window size** The advertised window size of the receiver is represented by a continuous line which grows in discrete steps. The distance from the ACK packet to the horizontal level of the line directly above it represents the actual window size.
8. **Notices** See requirement A3.
9. **Grid** The grid is drawn using horizontal and vertical lines drawn at regular intervals, which are dependant on the scaling used for the axes.

Requirement A10
Name Notes
Priority P1

The events have a similar display for user-specified notes as in the MSC. Notes must be distinguished from notices (aka. the "balloons"). Notices are described in A3.

Function A10.1
The Notes and Legend display shares the same space using tabbed panels. The text in the Note panel is specific to the active unit. During animation, the active unit is the latest drawn unit. The active unit can also be selected by clicking on the unit. The notes are user editable.

Requirement A11
Name Crosshair
Priority P3

A crosshair is to be added to the drawing area. The crosshair consists of a horizontal and a vertical line, which intersect over the higlighted packet.

Function A11.1
During animation, the latest drawn packet is highlighted, and the user can also choose the highlighted packet with the mouse by clicking on it.

3.2.2 Message Sequence Chart

The Message Sequence Chart (MSC) animation type is one of the two main animation views of the DaCoPAn2 Animator. It is based on the MSC animation panel, see Figure 5, of the original DaCoPAn project; it shows application, network and transport layer message sequences in "text-book" style, as lines from one host to another. On both sides of the drawing area there will be two optional columns where timestamps and other information regarding the events is displayed.

The main difference between the MSC panels in DaCoPAn and DaCoPAn2 is that in the latter there is an alternate animation mode. The added animation mode, in Figure 6, no longer conforms to a true linear time scale on the y-axis. The new animation mode and changes to the current version will result in an improvement to the readability and functionality of the MSC panel. The new animation mode will be added to complement the already existing animation panel and not to replace it.

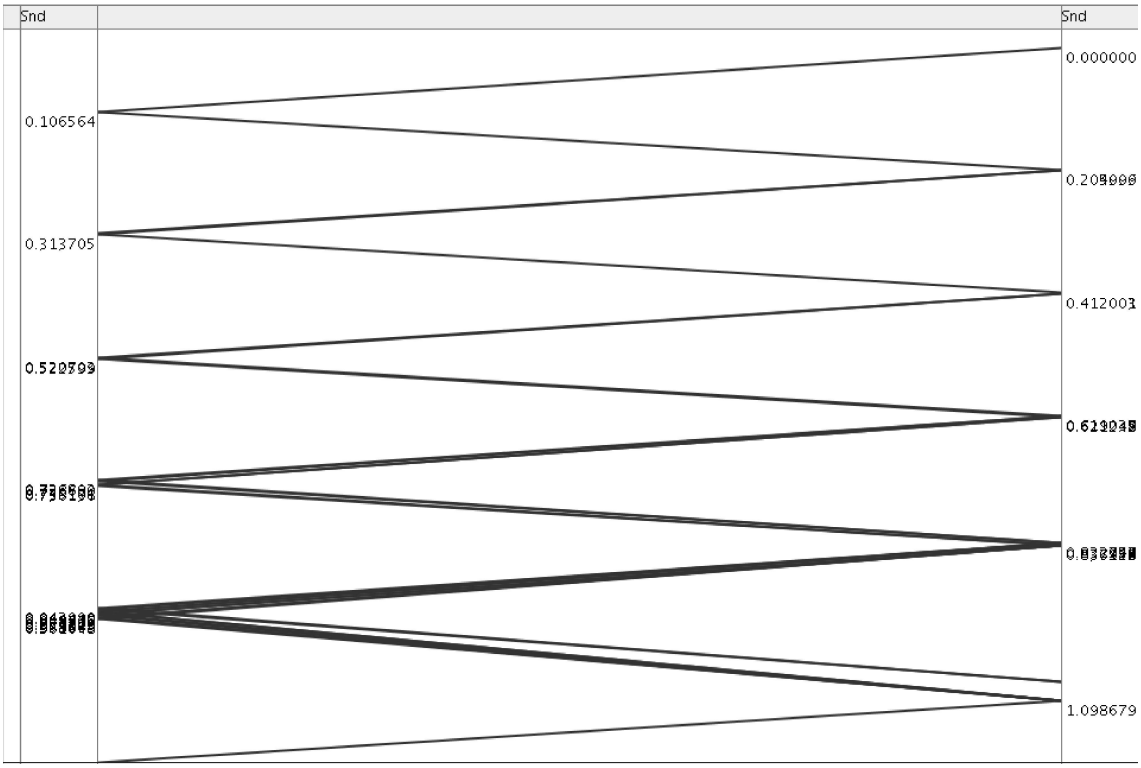

Figure 5: Message Sequence Chart with linear time scale

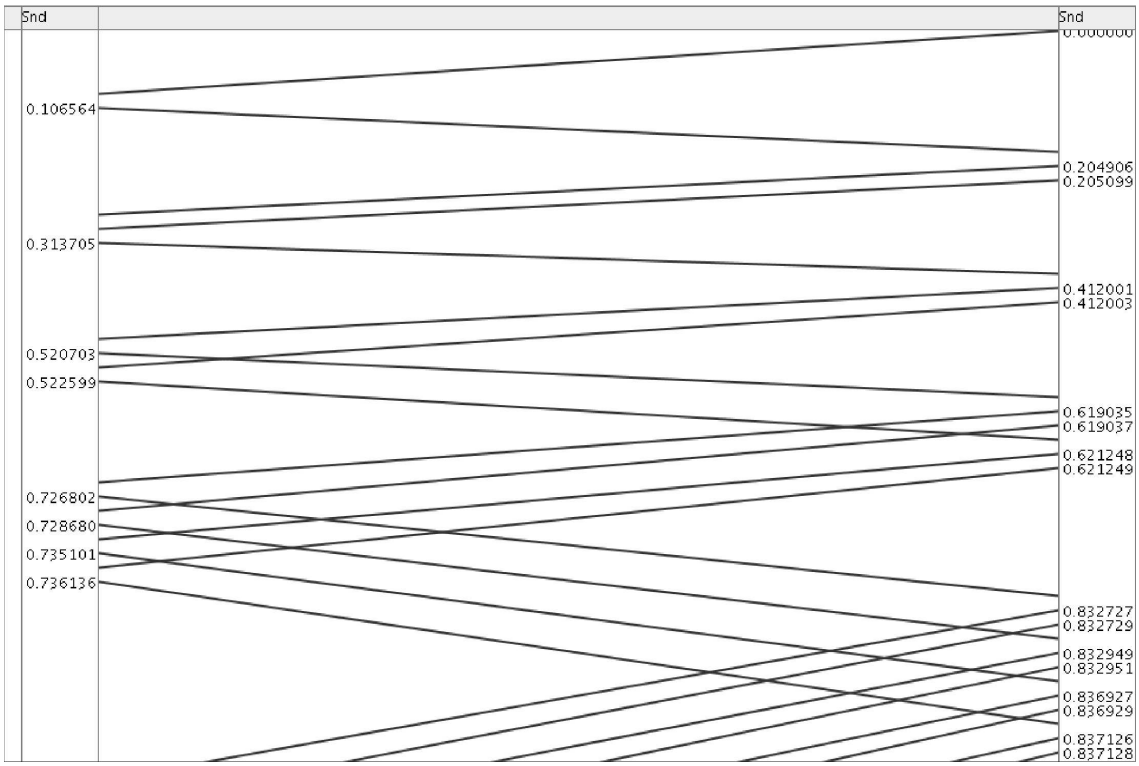

Figure 6: Message Sequence Chart with non-linear time scale

Requirement B1
Name Progress line scrolling
Priority P2

Moving the progress line up or down with the mouse, via click and drag, past the drawing area window borders should cause the display window to scroll.
In the present implementation, the user has to stop dragging the progress line and use the scroll bar to scroll the animation to the next screenful of events.

Requirement B2
Name Progress line snapping
Priority P4

The progress line should snap to the next event when dragging it with the mouse, while the animation is paused. This will aid the user in selecting active units.

Note: Could prove difficult to implement due to the code.

Requirement B3
Name Maximum distance of consecutive events
Priority P1

Two or more consecutive events should fit into the drawing area, if they are seperated by intervals, several screen lengths, in which no other events transpire.
This will fix the problem in the original MSC implementation where a major portion of the animation consists of the progress line scrolling down several empty screens of empty white space.

Function B3.1
Sections of the drawing area where there are no transpiring events are cropped to a certain length. If there are no sent or received units during a time interval, which has a length equal to the vertical height of the animation panel, then this empty time/space is removed.

Requirement B4
Name Minimum distance of consecutive events
Priority P1

All events should be distinctly distinguishable from each other in the drawing area. For example, adjacent timestamps shouldn't overlap.
This will improve the situation in the original MSC where several units which are transferred almost simultaneosly are drawn on top of each other so they appear to the user as one unit. Also, the text written on top of these unit lines overlap each other and are unreadable.

Function B4.1
A minimum spacing/distance, referred to as "min stepping", is to exist between adjacent events/units. The spacing should, by default, be of a sufficient length so that the header variables above the line, timestamps and column variables are clearly distinguishable.

Requirement B5
Name Gradients
Priority P1

The gradient of a line depicts the transition time from one host to another. The gradient of a line should be comparable to other lines.

Function B5.1
The gradient of a line is limited by a maximum spacing/distance adjustment, referred to as "max stepping". This is to enable the line to be fit into the drawing area, if the minimum stepping of any other lines doesn't cause the line to be scaled to a greater

length than the maximum stepping. Thus, “max stepping” is always secondary to “min stepping”. If they conflict then the “max stepping” is to be stretched instead of shrinking the “min stepping” for any lines. As a result, this may lead to the disproportionate scaling of the gradients for certain lines.

Requirement B6
Name Horizontal lines
Priority P2

Straight horizontal lines are allowed, if the unit direction is distinguishable.

Function B6.1
The lines will have arrow heads which distinguishes the direction of transmission.

Requirement B7
Name Variable font-size
Priority P1

The font size for header information appearing above the drawn lines should be adjustable to improve the readability on different display set-ups.

Function B7.1
Adjusting the minimum stepping of a line (see B4) should also automatically adjust the font size, or such an option should be available in the settings.

Requirement B8
Name Variable columns
Priority P1

Unit variables can be displayed in a column next to the timestamps. In the present implementation, variables are drawn only on top of the unit lines, so that the line length restricts the amount of shown variables.

Function B8.1
There are to be two columns per host, of which one has a timestamp value and the other contains unit variables which are seperated by captions. Changing the number of variables should also adjust the “min stepping” (see B4).

Requirement B9
Name Host fields
Priority P1

The host fields located on top of the drawing area are to be removed. This will add more vertical space for the actual animation.

Requirement B10
Name Send and receive time differentiation
Priority P2

The send and receive times of drawn lines should be clearly distinguishable from each other, so that the transfer direction of the units can be seen.

Function B10.1
Color can be used to differentiate send and receive times from each other.

Requirement B11
Name Dropped units
Priority P1

Disappearing units cannot precede previously sent units. In the present implementation, disappering units too often intersect with other units making them hard to read.

Function B11.1
The gradient of a dropped unit should be the same as that of the nearest sent unit.

Requirement B12
Name Dropped units and intersecting lines
Priority P3

A disappearing unit cannot be drawn into an intersection of any other lines.

Requirement B13
Name Dropped unit line length
Priority P2

The line for a disappearing unit should be of sufficient length to allow header variables to appear above it, an improvement to the readability of the existing implementation.

Requirement B14
Name Scale settings
Priority P1

The scaling for the window should be adjustable without leaving the settings panel. Currently, the user has to spend a considerable amount of time to switch between the settings and animation windows to choose a fitting scaling.

Function B14.1
The settings panel should have an apply button, which updates the drawing area without having to exit the settings window.

Requirement B15
Name Non-linear time scale mode
Priority P1

A separate animation mode for the MSC is to be implemented, where the y-coordinate of the drawing area no longer follows a true linear time progression. This is to complement the already existing true linear time scale animation mode, which has the problem of overlapping units as described in B4.

Function B15.1
The settings panel should have an option to use a true time or non-linear time scaling for the y-axis. The use of the non-linear time scale enables the use of the min/max stepping option (see B3 and B4), whereas the true time scale disables this feature.

3.2.3 Encapsulation

The requirements for the Encapsulation view are listed below, although they have been chosen not to be fullfilled during the project.

Requirement C1
Name Displayed headers
Priority P4

The header fields which are shown in the units have to be made customizable by the user from the settings.

Note: Not being implemented due to time constraints.

Requirement C2
Name Header and data sizes
Priority P4

The relative sizes shown for the unit headers and the data portion of the unit is disproportionate. A more intuitive way of displaying the data could be devised.

Note: Not being implemented due to time constraints.

3.2.4 Unit Flow Orchestration

The UFO animation type displays the flow of transfer units as rectangular units traversing from one side of a small tube to the other. Since the MSC panel and the UFO panel occupy the same window, and are animated synchronously, changes made to the former will also have to be reflected in the latter.

Requirement D1
Name Overlapping units
Priority P3

Units should never overlap. This applies both to units sent at the same time and to units which bypass one another.

Function D1.1
The width of the unit is cut, compared to the existing implementation. The width of the unit does not correspond to original packet length.

Function D1.2
The minimum spacing implemented in the MSC view, described in B2.2, also affects the corresponding units in the UFO view. The x-coordinate of the unit in the UFO view is calculated from the intersection of the unit line and the progress line in the MSC view.

Function D1.3
Overlapping units are drawn next to each other in all cases.

Requirement D2
Name Emphasizing the active unit
Priority P2

The unit whose detailed information is displayed should clearly stand out from the others.

Function D2.1
The color of the active unit distinguishes it from the other units.

3.2.5 Settings

Listed below are the user requirements which are concerned with program settings in general. Settings specific to different animation types are found in their corresponding sections.

Requirement E1
Name Persistent system settings
Priority P2

User has to be able to save all global settings used in the program. Additionally, relevant settings stored in scenario files override the global settings. If there are no chosen and stored settings, the system will use predefined settings. The settings are user specific.

Function E1
The program maintains a global settings file. One is included in the distribution. The user has an option to save all settings. If no settings file is found, one will be generated with predefined values hard coded in the program.

Requirement E2
Name Animation type settings
Priority P2

Each animation type must have its own settings.

Function E2
Each animation type has its own tab in the settings panel. The active tab is chosen according to the animation type panel currently active in the main window.

3.2.6 General UI requirements

The following requirements are related to aspects of the user interface which are not directly related to any animation type described above.

Requirement F1
Name Text readability
Priority P1

All displayed text should be readable regardless of different system color schemes. The existing version has black text on a dark blue background on the main user interface, a combination which is indiscernible on the current Computer Science Departments Linux configuration.

Function F1.1
All the default color choices which are not user specified should be made so that related colors, for example text and background colors for a visual element, produce discernible combinations. If the text color code is hardcoded, so is the bacground color. If one is relative to the system color scheme, so is the other.

Requirement F2
Name Maximum animation panel area
Priority P2

The visual area for the animation panels should be as large as possible. The actual layout of the panels should waste as little space as possible.

Function F2.1
When changing the main window size, the sizes of the currently displayed animation panels are kept at reasonable values in relation to each other. For example, the MSC panel size is larger than the UFO panel size by default.

Requirement F3
Name Opening animator files
Priority P2

By default, PEF files in the current directory will have to be shown in the Open File dialog.

Function F3.1
The Open File dialog shows both supported file types, PEF and SCE files, in the current directory by default.

3.2.7 Miscellaneous requirements

This section describes the other functional requirements that are not covered by any of the other previous sections.

Requirement F4
Name Language corrections
Priority P3

A term in the Finnish localization has to be corrected: “siirto” is actually “kuljetus”.

Requirement F5
Name Unit payload
Priority P1

When viewing unit headers on the transport level, it has to be possible to display its payload.

3.3 Non-functional quality requirements

This section describes the desirable properties the produced software should have in order to satisfy the essential demands its typical uses will have. The fact that the project will potentially have further follow-up software engineering projects is also noted.

Requirement F6
Name Localization
Priority P1

The current level of localization will be maintained. New features will have to be localizable. The new features are impemented in English.

Requirement F7
Name Educational presentability
Priority P1

The main purpose of the DaCoPAn2 project is to maintain and enhance the educational aspect of the Animator software. This means that special importance is assigned to clarity of visual presentations. This will sometimes have to be achieved by simplifications and modifications of actual measured data; for example, in the minimum stepping distance feature of the MSC animation view (See B4). The software will have to be usable in class room context using a data projector as a display. This is reflected in the fact that the font sizes and colors used in the animations are user customizable (See B7, A7).
Function All

Requirement F8
Name Maintainability
Priority P1

The level of maintainability of the original software must not deteriorate, anticipating possible future follow-up projects.
Function All

Requirement F9
Name Compatibility with Analyzer
Priority P1

The Animator must remain compatible with the PEF file specification in [1]. No modifications to the PEF specification will be made, since all the new features can be added via its extension support.

References

1 DaCoPAn Software Engineering project, *Design*. Release 1.0. Universities of Helsinki and Petrozavodsk, April 2004.

2 DaCoPAn Software Engineering project, *Requirements specification*. Release 1.0. Universities of Helsinki and Petrozavodsk, March 2004.

Appendix A. Glossary

The following are definitions, acronyms and abbreviations used in this document.

ACK packet: ACK packets are used to acknowledge the receipt of a packet in the Transmission Control Protocol (TCP). They are used by both ends of the connection to move in between states, and are the basis of TCP's reliability.

Analyzer: A module of the DaCoPAn project. It reads and analyzes packet trace files and produces a protocol events file (PEF) as its output.

Animator: A module of the DaCoPAn project. It reads protocol events files (PEF) and scenario files (SCE files), and animates the protocol exchanges using various parameters. Its main use is as a teaching tool to instruct students on protocol basics found in the protocol event file.

balloon: See **notices**.

DaCoPAn: DaCoPAn stands for visualization of Data Communication Protocol through Animation.

disappearing unit: See **dropped packet**.

dropped packet: A unit which has vanished during the course of network transmission.

drawing area: The section of the MSC and TSC animation views where the actual network data is visualized and animated.

ENC: An abbreviation for the Encapsulation view. The Encapsulation (ENC) panel is one of four animation views which the software can display.

event: The occurance of a protocol exchange. For example, an event can be the sending or receiving of a unit.

header variable: Various TCP or other network layer variables which are contained in the transfer unit header fields.

MSC: An abbreviation for the Message Sequence Chart view. The Message Sequence Chart (MSC) panel is one of four animation views which the software can display.

notices: Notices are automatically generated by the TSC view based on various events in the PEF file and displayed inside small rectangles in the notice bar.

notes: Notes are free format textual information that can be added, edited and removed freely by the user. Both the MSC and TSC animation views contain a separate notes panel.

PEF: The abbreviation stands for Protocol Events File, and is synonymous with the term PEF file and PEF files, as used throughout the document. It is an output file generated by the Analyzer and read by the Animator. It contains data about the network traffic that will be animated by the Animator.

progress line: The blue horizontal line in the MSC panel, which scrolls down the view as the animation proceeds and reveals transpiring events as they unfold.

SACK: Selective Acknowledgment. An optional TCP feature which allows the receiver to specify which segments it has received and which ones require retransmission.

scenario file: A native file written and read by the Animator. It contains all the data from a protocol events file and additional data related to how the scenario can be presented to the user, including notes and breakpoints.

SCE file(s): See **scenario file**.

sequence number: Every segment of data sent over a TCP connection has a sequence number. The sequence number is the number of the first data byte in the segment.

SEQNO: See **sequence number**.

TSC: An abbreviation for the Time Sequence Chart animation view. The Time Sequence Chart (TSC) panel is one of four animation views which the software can display.

UFO: An abbreviation for the Unit Flow Orchestration view. The Unit Flow Orchestration (UFO) panel is one of four animation views which the software can display.

unit: A transfer unit, which refers to all logical units that are transferred regardless of network layer. Thus, a unit can be an IP packet, a TCP segment or an application layer message.