Project plan

DaCoPAn

Helsinki 18th February 2004

Software Engineering Project

UNIVERSITY OF HELSINKI
Department of Computer Science

UNIVERSITY OF PETROZAVODSK
Department of Computer Science

Course

581260 Software Engineering Project (6 cr)

Project Group

Carlos Arrastia Aparicio
Jari Aarniala
Alejandro Fernandez Rey
Vesa Vainio
Jarkko Laine
Jonathan Brown

Kirill Kulakov
Andrey Salo
Andrey Ananin
Mikhail Kryshen
Viktor Surikov

Customer

Markku Kojo

Project Masters

Juha Taina (Supervisor)
Yury Bogoyavlenskiy (Supervisor)

Turjo Tuohiniemi (Instructor)
Dmitry Korzun (Instructor)

Homepage

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

Change Log

Version	Date	Modifications
1.00	17.02.2004	The first published version.

Contents

1 Introduction

DaCoPAn is a joint, distributed software engineering project between the University of Helsinki, Department of Computer Science, and the University of Petrozavodsk, Department of Computer Science. The students participating the project are representatives of three universities: Helsinki, Petrozavodsk, and Autonoma of Madrid. The name **DaCoPAn** stands for visualization of **Da**ta **Co**mmunication **P**rotocols through **An**imation.

The general goal of the project is twofold (in order of priority).

1. Experimenting with a geographically distributed software engineering project. The idea for this type of inter-departmental cooperation was formulated in [1].
2. Developing a simple yet extensible software program for visualizing the behavior of data communication protocols through animation.

The students receive an impression of both a typical software engineering project and of a distributed one. The project stakeholders gather and analyze information about these types of projects. The customer receives the software which implements basic functionality and supports future extension.

The general idea of the desired software is playback animation of packet trace information captured from real data communication traffic. Animating TCP/IP traffic is the priority. The packet traces are gathered at each end point participating the communication; the basic case is two end points and two tcpdump output logs. The problem of gathering is not a part of the software to develop.

The aim of the software is thus to help in study and analysis of the protocol message exchange by animating its behavior. This covers both educational and research targets but the latter has less priority.

The software is divided into two subsystems in accordance with two major required functions, see Figure 1. These subsystems can run both as an integral system and as standalone applications.

The first subsystem is an analyzer of tcpdump packet traces, one for each end point, perhaps with some additional data. The traces are combined to create a complete view of the events in the chronological order. An extensible intermediate format has to be designed and used to store the combined data (protocol events file).

The second subsystem is an animator of protocol events collected with the analyzer. The main form is the chronological drawing using a message sequence chart. Other types of animation can be considered as well. Different layers of the Internet protocols stack should be supported. A user is also allowed to configure and tune the animation.

The general rules for the project to follow are stated in [2]. The project starts 21.01.2004 and ends 31.05.2004. There are only two periods when both groups work together in a non-distributed manner: starting the project (Helsinki, 26.01–10.02.2004) and integration testing (Petrozavodsk, 03–13.05.2004).

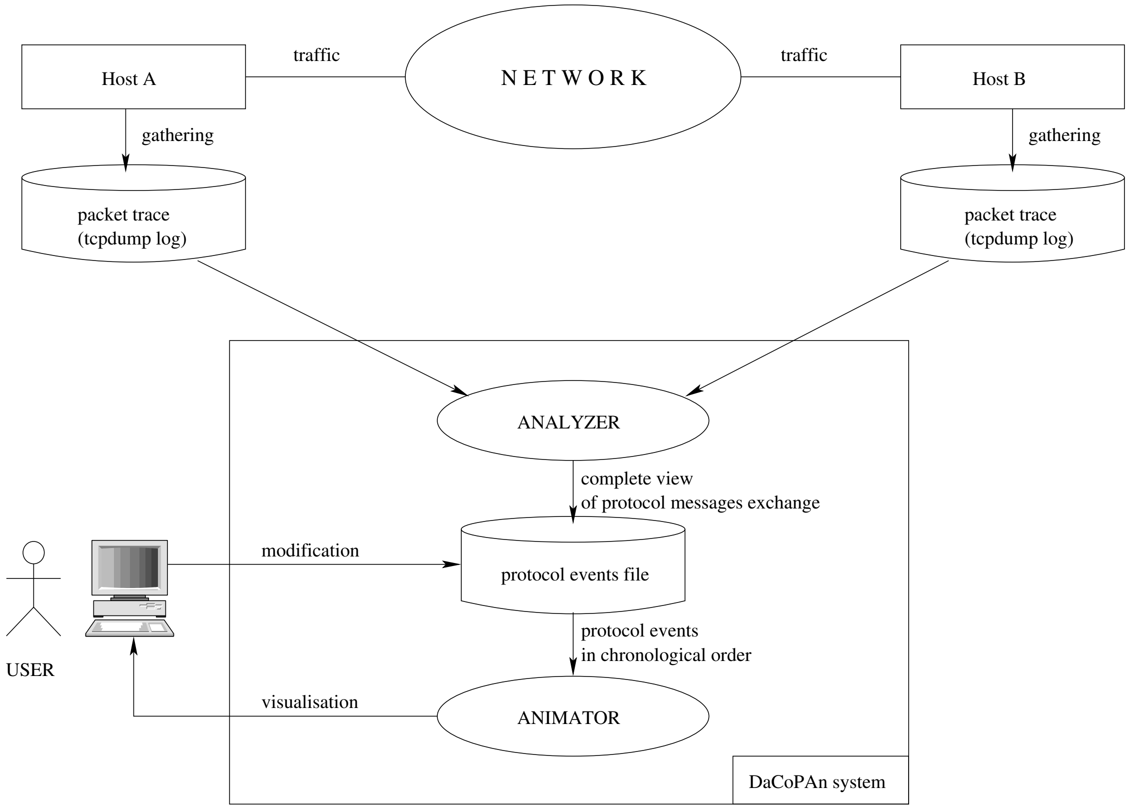

Figure 1: The basic DaCoPAn system context

2 Organization

The DaCoPAn project team is divided into two groups: one group works in Helsinki, the other one works in Petrozavodsk. The general structure of the project organization is shown in Figure 2.

TWiki website:
`http://db.cs.helsinki.fi/~tkt_daco/twiki/bin/view/Main/DaCoPAn`

CVS repository:
`cs.helsinki.fi`: `/home/group/dacopan/`

Mailing list of the global team:
`ohtuk04-dacopan-global@cs.helsinki.fi`

Forum:
`http://db.cs.helsinki.fi/~tkt_daco/cgi-bin/forum/YaBB.cgi`

2.1 Supervisors

Juha Taina, Senior Lecturer of Computer Science Department of the University of

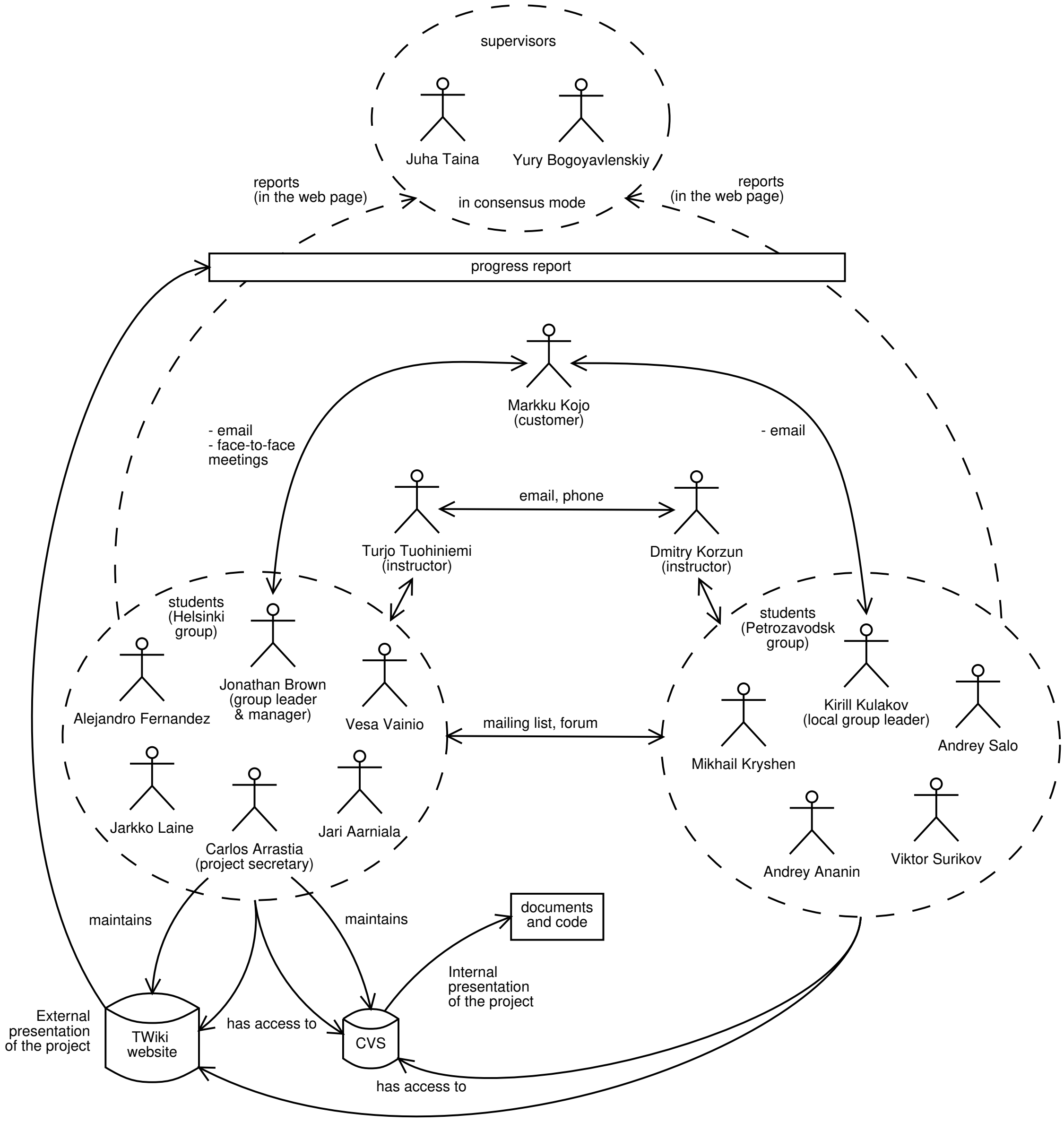

Figure 2: The DaCoPAn project structure

Helsinki, PhD.

Phone: +358 9 191 44226
Fax: +358 9 191 44441
Email: juha.taina@cs.helsinki.fi

Yury Bogoyavlenskiy, Head of Computer Science Department of the University of Petrozavodsk, PhD, associate professor.

Phone: +7(8142)711015
Fax: +7(8142)711000
Email: ybgv@cs.karelia.ru

2.2 Customer

Markku Kojo, Senior Researcher of Computer Science Department of the University of Helsinki.

Phone: +358 9 191 44179
Fax: +358 9 191 44441
Email: markku.kojo@cs.helsinki.fi

2.3 Instructors

Turjo Tuohiniemi, Lecturer of Computer Science Department of the University of Helsinki.

Phone: +358 9 191 44157
Email: turjo.tuohiniemi@cs.helsinki.fi

Dmitry Korzun, Senior Lecturer of Computer Science Department of the University of Petrozavodsk, PhD.

Phone: +7(8142)711015
Email: dkorzun@cs.karelia.ru, korzoun@cs.helsinki.fi

2.4 Helsinki student group

Mailing list of the Helsinki group:
`ohtuk04-dacopan-helsinki@cs.helsinki.fi`

Jonathan Brown, group leader & project manager.

Phone: +358 50 5410112
Email: jonathan.brown@cs.helsinki.fi

Carlos Arrastia, website and CVS repository manager.

Phone: +358 50 4715104
Email: arrastia@cs.helsinki.fi

Jari Aarniala, code manager.

Phone: +358 40 5394434
Email: jari.aarniala@cs.helsinki.fi

Alejandro Fernandez, quality assurance and testing manager.

Phone: +358 50 4719079
Email: afernand@cs.helsinki.fi

Vesa Vainio, educational specialist & UI manager.

Phone: +358 40 5636966
Email: vesa.vainio@helsinki.fi

Jarkko Laine, documentation manager.

Phone: +358 400 778508
Email: jarkko.laine@helsinki.fi

The secretary duties in meetings are rotated between group members.

2.5 Petrozavodsk student group

Mailing list of the Petrozavodsk group:
`ohtuk04-dacopan-petrozavodsk@cs.helsinki.fi`

Kirill Kulakov, group leader, documentation manager.

Master Student of Computer Science Department of the University of Petrozavodsk.

System analysis, modeling, quality assurance, management.

Email: kulakov@cs.karelia.ru, kulakov@cs.helsinki.fi

Andrey Salo, code & technical manager.

Senior Student of Computer Science Department of the University of Petrozavodsk.

Java programming, C programming, translators development, distributed systems.

Email: salo@cs.karelia.ru, ansalo@cs.helsinki.fi

Andrey Ananin, design manager.

Junior Student of Computer Science Department of the University of Petrozavodsk.

C programming, UML design, data structures and algorithms.

Email: ananin@cs.karelia.ru, ananin@cs.helsinki.fi

Mikhail Kryshen, testing & quality assurance manager, keeper of Helsinki group activity track.

Junior Student of Computer Science Department of the University of Petrozavodsk.

User interface design, architecture, Java programming, Web-design, C programming.

Email: kryshen@cs.karelia.ru, kryshen@cs.helsinki.fi

Viktor Surikov, participating the project since Feb 16 2004, developer.

Junior Student of Computer Science Department of the University of Petrozavodsk.

Email: surikov@cs.karelia.ru

The secretary duties at the local meetings are rotated between Andrey Ananin, Mikhail Kryshen and Andrey Salo.

2.6 Explanation of the roles

In the DaCoPAn project group the following roles have been created, defined and assigned. These roles are roles designed to assign leadership and responsibility, not accountability. Each group will make sure that every task is completed as a combined effort, with the corresponding manager as a driving force, making sure that the effort is focused and on target. At no point should this leadership surpass the obligation to a democratic, fair and reasonable decision making process within the project group.

Project manager

This person will take a higher level responsibility for the flow of the project. He will strive to act as a chairman at any meetings where it is not otherwise predefined who will lead the discussion. He we try to keep meetings on track and act as an outside spokesperson wherever the project group needs one. Reporting project progress should be orchestrated by this person.

Group leader

This person will lead his group to achieve the task at hand. If any question exists about the sharing of the burden of a particular task, this person should decide the division of work.

Code Manager

This person supervises the correct and proper development of code in his project group. He makes sure that coding standards are observed, and that the code is commented, clear and well designed.

Documentation magager

This person sees that documents are available to all relevant parties and that documents are created with the proper structure. Assisting the rest of the group with practical issues concerning documentation is one of the important responsibilities of this person.

Quality assurance manager

This person assures quality of the software and the software production process as well. This person will also plan and lead the effort to localize the software into the 4 languages that are planned to be supported at release time.

Testing manager

This person leads the construction of the test plan and test cases, leads the testing itself and helps coordinate the documentation of the testing.

UI manager

This person is active in the design phase as a user interface designer, and then coordinates the effort of implementing the UI with as much quality as possible. Also helping the group to arrive at the correct UI design choices will be key in this persons contribution to the project group.

Educational specialist

This person helps the group to define and then achieve the educational goals of the software. Helping to construct the user interface and making sure that principles of effective learning are observed in the design phase are very important jobs of this person.

Website and CVS manager

This person enables and facilitates the operation of the group in its combined effort to create the software using the distributed project model by maintaining and updating the group website and taking care that the CVS system is properly structured and properly maintained.

3 Working procedures, monitoring and reporting

The procedures used in the day-to-day operation of the project are described in this section to make it as easy and straightforward as possible for the group members to contribute to the project. Because of the distributed nature of this project communication is the most critical issue and therefore this section focuses primarily on this aspect.

3.1 Communication

In the DaCoPAn project communication happens through five channels. Each has a distinct advantage for a different tasks: Email, mailing lists, a Team Wiki (Twiki) web system, a forum and meetings are used. In some cases it might also prove useful to use instant messaging systems like ICQ or MSN Messenger. Using a telephone or other methods of communication are also possible in special cases.

3.1.1 The mailing lists

There are three different mailing lists:

- ohtuk04-dacopan-global@cs.helsinki.fi for global issues
- ohtuk04-dacopan-helsinki@cs.helsinki.fi for the Helsinki group's internal issues
- ohtuk04-dacopan-petrozavodsk@cs.helsinki.fi for the Petrozavodsk group's internal issues

The mailing lists should be used mainly for informing other group members, as it works best with messages that don't form long reply threads. For time critical questions mailing lists are also the best means of communication.

3.1.2 The project discussion board

The discussion board located at tkt_daco/cgi-bin/forum/YaBB.cgi on http://db.cs.helsinki.fi/ is the main channel for discussing specific aspects of the project. It suits longer message threads better than the mailing lists because it stores the messages in a well organized structure. It is therefore suggested that all messages that are likely to have more than three replies and are not extremely time critical should be sent to the discussion board instead of the mailing lists.

If the messages added to the discussion board are time critical and require fast reply, a note about the created message can be added to the mailing list as well.

Group members should follow the board actively, both by visiting the forum and by using the "Notify of replies" function of the bulletin board whenever they feel that it is necessary.

3.1.3 Email

In addition to the mailing lists group members can send each other email messages in case they just want to ask small questions from some other project member or comment on his work. In any case all larger issues that can effect other people's work as well should be addressed to the whole group or subgroup through the mailing list or discussion board.

In email messages it should be clearly stated in the first text line how soon and from whom a reply is expected. If this is not stated, it is supposed that the message is not time critical and everyone can use their common sense to decide whether to reply to the mail or not.

3.1.4 Meetings

Both subgroups should organize local meetings where they meet face to face to discuss the project. Even though there are many other communication methods as well, the power of "real" communication should not be underestimated. Because of the distributed nature of this project it is not usually possible for the whole group, meaning both subgroups, to meet together, so it is very important that the groups write meeting minutes about their formal meetings. By reading these minutes the groups can follow the meetings held by the other group.

In addition to formal meetings the groups or some parts of the groups can meet informally and discuss project related issues. The ideas that come up in informal discussion should be also taken into account and documented with some other form of communication, which can be for example writing an email or bringing it up in a formal meeting.

3.1.5 Instant messaging

Instant messaging is not to be considered as an official communication method for the project since it is more like informal than formal discussion. It yet has its advantages and should therefore be taken into consideration in some special cases.

Because of the distributed approach it is not possible for members of two different subgroups to meet each other face to face — this is where instant messaging can help. For example when two members of two different subgroups are developing parts of the software it might be useful for them to be able to discuss what they are doing in real-time. This cannot be done in a simple way by any of the means described above, but it is possible using instant messaging.

3.1.6 Communication with the customer

Communication with customer is handled by two methods: e-mail and forum. E-mail is preferred for single questions that can be probably answered with just one or two messages. When e-mail is used, it is important that the information from the customer reaches the whole group. Notifying other group members is the responsibility of the person communicating with Mr. Kojo. If no reply is received from the customer after two or more days, the author of the message can poll by sending another message asking for reply.

The discussion board (forum) can be used in cases when it is probable that the discussion will require more than one or two messages and it is useful to store the messages as a thread in the forum. In this case the customer needs to be informed by e-mail that there is a message waiting for him on the forum. On the forum all communication between the customer and the project group should take place on the section “Customer communication”.

The most important thing in communicating with the customer is that he does not have to discuss the same things many times with different people. So it is important that the whole group is informed on discussion that has taken place between Mr. Kojo and some group members.

3.2 Cooperative work

The DaCoPAn project — because of its distributed nature — relies even more than other projects in strong and seamless cooperation between all the participants of the project. One part of this is the communication discussed earlier in this section, and the other part is sharing the documents and other artifacts produced in the course of the development work.

3.2.1 CVS repository

The sharing of documents is mainly done via a CVS repository, in which documentation and programming language code is stored in text format. Each member of the project should check the repository every time before starting working on the project and update their work to it as often as it is reasonably possible.

Any project member can make changes to the documents created by other members — but if the changes are somewhat bigger than just changing a word, and it is not clear that the original author would agree on the changes, he should be contacted by e-mail prior to making the change.

The repository is organized in the following way:

Structure

- Documents: The documents created in the course of the DaCoPAn project including the working hour logs from the project members
- Source: The project source code is divided into two folders: animator and analyzer.

3.3 Monitoring and reporting mechanisms

An additional demand for the working procedures is that the supervisors and the customer should be kept up to date with the state of the project. This is achieved through different means of monitoring and reporting discussed in this section.

3.3.1 Monitoring

The communication between DaCoPAn group members should be conducted in a transparent way using the methods presented in the previous section. The instructors follow the mailing lists, forum and TWiki, so monitoring the communication between group members is quite easy. Everything other than communication can be monitored through the TWiki by any person interested in the project.

Beside that, following actions have to be taken to make it possible for the supervisors and customer to get a complite picture of the project.

- Minutes of the meetings must be taken for every meeting (global and internal) using the template available in TWiki (MeetingMinutes).
- During the course of the project several documents have to be published on the project TWiki. In addition to that the instructor can request the latest versions of the documents directly from the documentation managers. The documents published during the project are:
  - Project plan (this document)

  - Requirements specification
  - Design document
  - Test plan
  - Implementation document
  - Well commented C and Java source code
  - Test execution document
  - Conclusion
  - User manual

- Internal memos can be written for aspects that are not covered enough by other documents. These memos can be then added to the TWiki.

3.3.2 The TWiki web system

The web system (tkt_daco/twiki/bin/view/Main/DaCoPAn on http://db.cs.helsinki.fi) is used mainly for storing finished versions of the documents created in the project and to present the project to visitors from outside. In this use it works like a normal web page except that all group members can easily update the files in it.

In addition to the main purpose described above the web system also serves as a means of sharing useful project related information between the group members by adding links or small documents to the system. Every section in TWiki can be edited and updated by any group member.

Structure:

- Home: Introduction / Current status / History of the project and news.
- Overview: Project description / Participating universities / Project Goals.
- Members: Member profiles / Mailing lists.
- Documentation: Document templates / Minutes of the meetings / Time reports template.
- Resources/Links: TCP protocol information / Tools used / TWiki resources / Other links.
- Forum.

3.3.3 Reviews and oversights

Formal reviews are an important part of well-organized quality assurance as they provide a means to spot errors in an early phase of development. It is well known that a good review meeting is a much more effective tool for finding errors than even the best testing can ever be.

A review concentrates on one piece of programming code or documentation that is discussed based on preparations done in advance by attendants. Each member attending a review should thoroughly read through the product to be reviewed and write down their comments. A checklist can be used to help finding errors. Careful preparation assures that the meeting will not be longer than two hours, and the meeting stays effective.

In a review, the person responsible for writing the documents to be reviewed presents them and then the review leader (Quality Assurance manager) leads the conversation by walking through the documents. Each participant then presents his notes at the appropriate time.

It is important to note that in a review the point is not to find solutions for problems but just to spot errors. The producer probably will be able to fix them later by himself. Another important thing is that debate should be kept to the minimum and if such issue raises that the attendants cannot agree on, it should be left open for further discussion or the producer to decide.

In a formal review notes containing the following information should be taken:

- What was reviewed?
- Who reviewed it?
- What were the findings and conclusions?

At the end of the review the reviewers decide whether the reviewed product should be (1) accepted without further modifications, (2) rejected due to severe errors — once the errors have been fixed a new review must be performed — or (3) accepted provisionally, so that when the errors have been fixed no new review is needed.

The first product to be reviewed in the course of this project is the requirements specification. In this review the customer will also be present. After that a number of internal reviews can be held. The final amount of reviewing will be decided later.

3.3.4 Reporting

The project group should report to its supervisors by sending the following reports:

- **Progress report** Every member collects a report of his working hours, using the format description available in TWiki (WorkHoursTemplate) and uploads it to the CVS repository. These reports are then compiled to one progress report that is sent to the supervisors. Both subgroups create their own reports. An existing Perl script that creates this report based on the time reports of each member is available on CVS.

  This report basically indicates the hours worked by each member, the started project phases, the finished documents, and other issues and feedback to the management.

- **Conclusion** An analysis of the work completed and self-evaluation document must be written at end of the project.

4 Resource requirements

4.1 tcpdump

The tcpdump packet traces described in the Network Communication Scenarios document are needed for creation and testing of the analyzer program. The customer should provide with these as soon as the scenarios are defined.

For extra information it might be useful to have access to run tcpdump by the team itself as well so that they could for example create non-standard test cases.

4.2 Similar existing programs

Examining some examples of existing TCP-tracing tools might be useful in seeing how other people have dealt with similar problems. For example running Sea Lion or Sea Wind software might be interesting.

4.3 Development and testing tools

Common tools for modelling must also be available, as well as appropriate tools for software development and testing.

4.4 Working environment

Both groups should have access to several environments to develop and test the program, or at least agree on a common operative system.

5 Size estimate

The following two techniques are used for the estimate: decomposition of the software and estimation by analogy.

Decomposition is very natural for this project because the team is divided into two groups (Helsinki and Petrozavodsk) as well as the software is an integration of two subsystems (Analyzer and Animator). Each subsystem is assigned to one group. Each group makes the estimation of its subsystem's size.

For the estimation by analogy, some similar but completed software in the same application domain are used.

5.1 Analyzer (Petrozavodsk group)

The Petrozavodsk group consists of 4 implementors; Kirill Kulakov as a group leader will not participate in pure coding. We assume the implementation phase has not to exceed four weeks and each developer can write at most 500 lines/month of effective code in ANSI C.

Therefore, the upper bound for the estimated total size of the analyzer is

$$4 \text{ persons} \times 1 \text{ month} \times 500 \text{ LOC} = 2000 \text{ LOC} .$$

The following software programs were considered for the estimation by analogy.

- `tcptrace` by Shawn Ostermann, Ohio University,
  `http://jarok.cs.ohiou.edu/software/tcptrace/tcptrace.html`

  `tcptrace` is a tool written by Shawn Ostermann at Ohio University, for analysis of TCP dump files. It can take as input the files produced by several popular packet-capture programs, including tcpdump, snoop, etherpeek, HP Net Metrix, and WinDump. `tcptrace` can produce several different types of output containing information on each connection seen, such as elapsed time, bytes and segments sent and received, retransmissions, round trip times, window advertisements, throughput, and more. It can also produce a number of graphs for further analysis.

- `tcpdump2xml` by Vadim Ponomarev, the University of Petrozavodsk, not in public domain.

  `tcpdump2xml` program is a tool written by Vadim Ponomarev at Petrozavodsk State University for tcpdump log files to XML format conversion. It is used to extract ethernet, ip and tcp headers from tcpdump log and write information to file in XML format. Format itself is developed by author.

These analogs are used to compute the proportion between the system modules of the analyzer. According to `tcptrace` and `tcpdump2xml` code analysis, 400 LOC is quite enough to implement a log reader. The message mapping problem seems to be more complicated than events calculating, thus more LOC are going to be used for message mapper implementation. The rest of the code is intended for protocol events file writing routines. The results of the analyzer size estimate are presented in Table 1.

5.2 Animator (Helsinki group)

Here is a description of the different subareas of the animator program that are used in the integral estimate.

Note: In the design phase it can be considered if it is feasible to create an animation framework (maybe with a Java interface) for expressing the common functionality between different animation types. This framework would provide abstraction for the time-related controls of the animations. The contract of the animations would concentrate on mapping these external control signals to the visual presentation of the animation.

- **XML to data structures** is a component that somehow reads or maps the XML data in the intermediate file format to internal memory data structures of the animator program. Some libraries will be used for low-level parsing of XML. There are suitable basic libraries in Java 1.4 SE, but another library can be used if a better alternative is found. The mapping from XML file to data structures can optionally be bidirectional, so that there would be a mechanism to automatically save the internal data structures in the same XML format. However, only the reading part is mandatory.

- **Data structures** includes Java classes that describe the internal representation of the animation data. It is estimated that the size of code is rather large compared to the required effort, because the classes would have lots of fields and accessors that are all functionally very similar.

- **Animation type: MSC** means the Message Sequence Chart animation. This includes all the code that is needed to visually present animation data as a message sequence chart animation. The animation needs to be configurable for different layers and for presenting different header fields and host variables. The animation code also needs to provide accessors for controlling the animation (play, pause, step back, step forward etc.).

- **Animation type: Enc** means the animation for packet encapsulation. The animation should be able to present how data from upper layers is encapsulated as payload for lower layers. The animation should also be able to present fragmentation and de-fragmentation. The animation should have two modes, one for preparing a packet to send and one for interpreting the data in a received packet. The visual presentation may also have two views, one to present all the relevant layers and one to present more detail on one particular layer. Alternatively the more detailed presentation may present more animation, while the less detailed view would mostly be a static diagram. As for the animation data from the intermediate file, the encapsulation animation doesn't proceed in time. From network traffic's point of view the encapsulation and de-encapsulation happen instantaneously.

- **User interface** means all the user interface not contained in any other module. This means the main frame for the program (including menus etc.) and any other dialogs that are needed.

- **Managing settings** means internal representation of animator settings, including writing them on file in XML format and reading them from file. The student end user should be able to download one educational scenario in just one file, and he should be able to start viewing the scenario just by loading the one file and giving a Play command. To achieve this, there needs to be a file format for storing both the animation data and the scenario-specific animation settings in the same file. (Only the animator module needs to be concerned about this. The scenario file format is a superset of the intermediate file format produced by the Analyzer program.)

5.3 Integral estimate

Table 1 contains results of size estimates for analyzer and animator, and integral estimate in LOC.

Subsystem	**Programming language**	**LOC**	**Effort**
Analyzer		< 2000	
log reader	ANSI C (+ POSIX)	< 400	20%
messages mapping	ANSI C	< 800	40%
events calculator	ANSI C	< 600	30%
protocol events file format	ANSI C, XML	< 200	10%
Animator	Java, XML	< 6500	
XML to data structures	Java, XML	< 500	12%
Data structures	Java	< 1000	8%
Animation type: MSC	Java	< 1500	28%
Animation type: Enc	Java	< 1500	28%
User interface	Java	< 1500	16%
Managing settings	Java	< 500	8%
Total size estimate		< 8500	

Table 1: Estimated size of code and relative effort estimates for DaCoPAn system

6 Schedule

This section contains global project schedule and local schedules for each group.

6.1 Global phases

During the project the following artifacts are produced and delivered to the customer: Project plan, Requirements specification, User manual and DaCoPAn system (source code and executable program). Integration testing is started simultaneously by both groups. In addition to this global schedule each group has their own schedules for the design, implementation and unit testing phases, in accordance with the global schedule.

The global schedule is shown in Table 2. The corresponded GANTT diagram is shown in Figure 3.

6.2 Petrozavodsk group schedule

The Petrozavodsk group schedule is shown in Figure 4. The first draft of most important milestones and dates for Petrozavodsk group are presented in the global schedule.

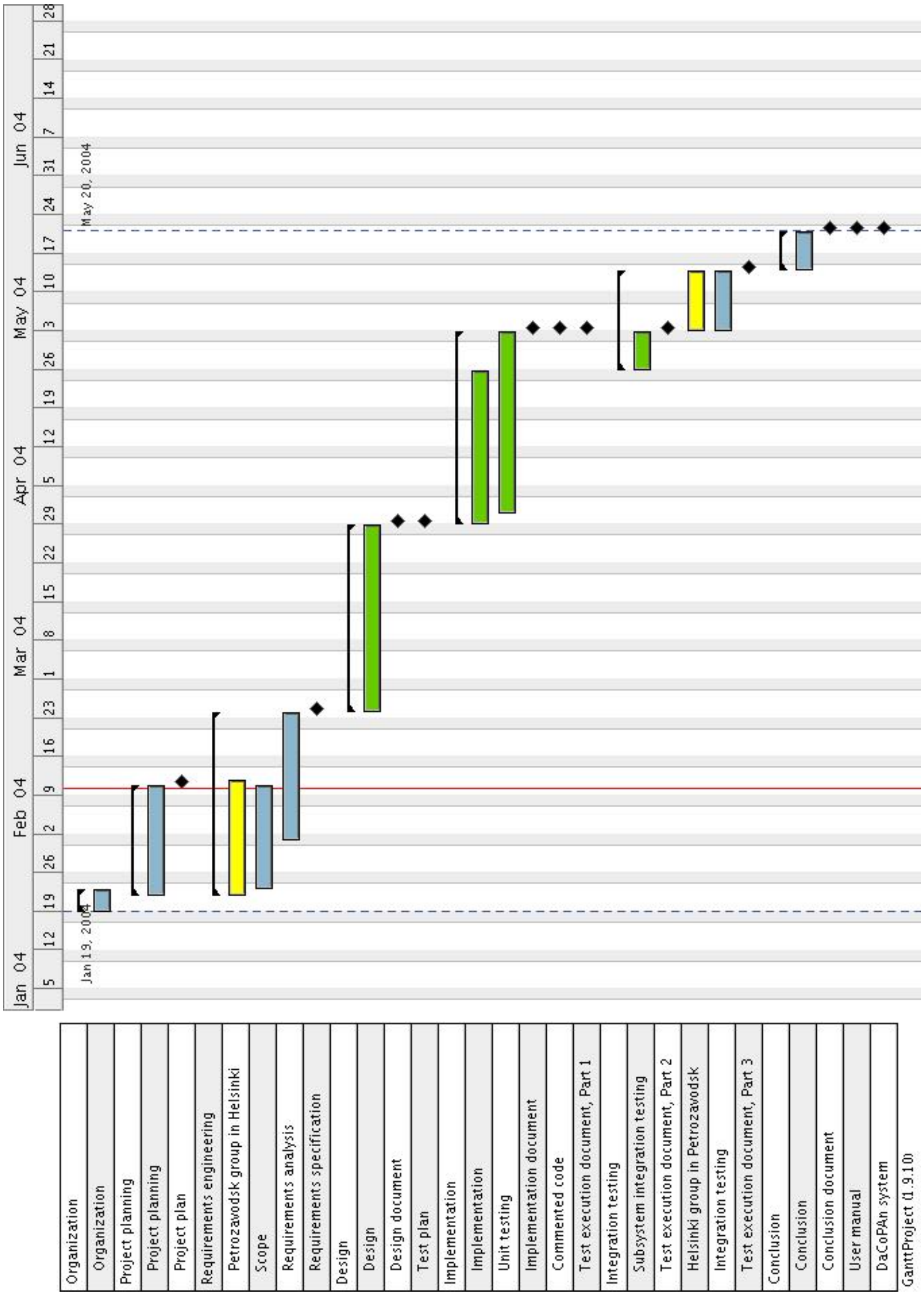

Figure 3: Global GANTT diagram

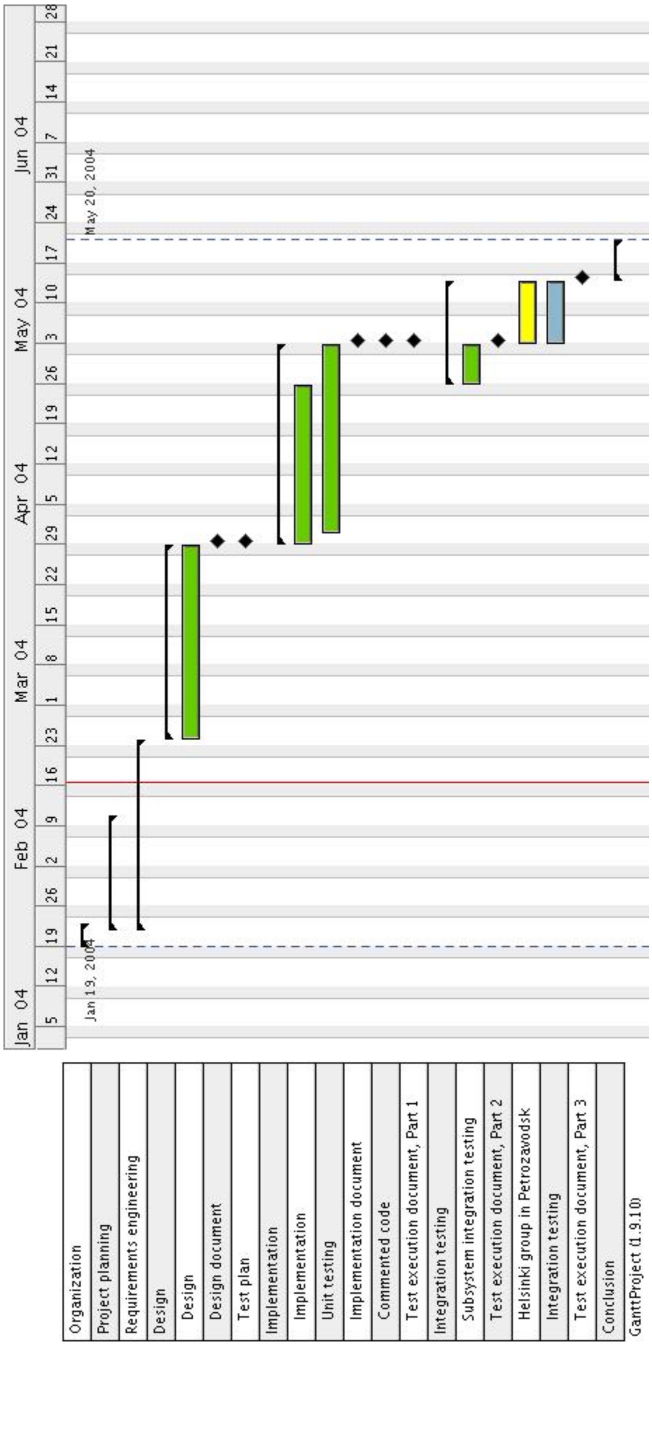

Figure 4: GANTT diagram for petrozavodsk group

Phase	Time period	Days	Produced artifacts
Project kickoff, organization	19.01–22.02.2004	3	
Petrozavodsk group in Helsinki	22.01–10.02.2004	20	
Scope, Requirement elicitation	23.01–10.02.2004	19	Requirements specification
Project planning	27.01–10.02.2004	15	Project plan
Requirement analysis	01.02–23.02.2004	23	Requirements specification
Design	24.02–28.03.2004	34	Design document, Test plan
Implementation	29.03–25.04.2004	28	Implementation document, Commented code
Unit testing	31.03–02.05.2004	33	Test execution document, Part 1
Subsystem integration testing	26.04–03.05.2004	10	Test execution document, Part 2
Helsinki group in Petrozavodsk	03.05–13.05.2004	10	
Integration testing	03.05–13.05.2004	10	Test execution document, Part 3
Conclusion	14.05–20.05.2004	7	Conclusion document, User manual DaCoPAn system

Table 2: Project schedule

6.3 Helsinki group schedule

Current consensus is that the internal schedule of the Helsinki group will adhere to the global schedule. All document milestones are expected to be met as specified in the global schedule.

7 Risk analysis

This section contains a breakdown of unforseen difficulties. Each risk that the project might face is documented, so it can be recognized at review meetings. There is an action plan of what to do if the risk materializes, and an estimate of the probability of the actualization of the risk. These probabilities can be graded as high, medium and low.

Risk	Short description of the risk
Probability	Low - Medium - High
Severity	Low - Medium - High
Minimizing risk	Steps to minimize the probability of the risk
Recognition	What are the signs of the risk materializing?
If materializes	What to do if the risk materializes, "damage control"

Table 3: Legend

Risk	The schedule of the project isn't met
Probability	Low
Severity	Medium
Minimizing risk	The project manager should monitor the progress of the project group and react if the project is lagging behind. Each member of the group should keep track of his own progress, and report to the project manager should his task(s) prove to be more time-consuming than was expected. The workload of the two groups should be balanced. Work should be distributed so that approximately the same amount of time is spent on the project per week.
Recognition	Milestones / checkpoints are delayed
If materializes	Rework the schedule so that the project can be finished on time, for example some parts of the project could be left out in order to deliver the product on time.

Risk	Incorrect distribution of development work
Probability	Low
Severity	Medium
Minimizing risk	Careful planning in requirement phase and in design phase
Recognition	If work load is not balanced, or if core skills required are not present in either group
If materializes	Some tasks can be reassigned and communication increased

Risk	Communication between groups insufficient
Probability	Medium
Severity	High
Minimizing risk	Maintaining established communication channels and making sure the content is correct.
Recognition	If there are inconsistencies or incompatibilities between the work of the 2 groups.
If materializes	Project managers re-establish the rules of communication and make sure all members are communicating the right information at the right times.

Risk	Members get sick or suffer other incapacitation
Probability	Low
Severity	Low
Minimizing risk	-
Recognition	-
If materializes	Some tasks can be reassigned and workload redistributed

Risk	Finished product doesn't meet customer's requirements
Probability	Low
Severity	High
Minimizing risk	Communicating with the customer efficiently during the requirements phase, also keeping close contact during the subsequent phases. Prototypes of the software could be shown to the customer before the product is finished in order to notice any inconsistencies with the requirements before the product is finished
Recognition	Customer is not satisfied with the final product
If materializes	If the risk materializes after the product has been finished, the chances of making any changes to it are minimal, since this project has a tight schedule. However, if prototypes of the program are shown to the customer during development and it is clear that the program will not meet the original requirements, some adjustments to the design / implementation of the program could be made.

Risk	The components of the product are incompatible
Probability	Low
Severity	Medium
Minimizing risk	Accurate specs of the interface between the 2 components
Recognition	If incompatibilities are encountered during integration testing
If materializes	Using specs, find out which component doesn't comply to the interface defined earlier. The non-compliant component should be adjusted so that it implements the interface.

8 SE techniques and CASE tools

8.1 SE techniques

The project will be carried out using the waterfall development model. This model is sufficiently simple, yet efficient for smaller projects, and considering that this is a first attempt at a distributed project, the model should be simple. Another reason for this model is that it limits the customer's presence mainly to the beginning of the process. His future availability for intense communication remains uncertain.
A use-case approach will be used for requirement elicitation and modeling user requirements (problem domain) and system requirements (software domain).

8.2 CASE tools

Table 4 shows a summary of chosen CASE tools for the DaCoPAn project.

8.2.1 Documentation

Documentation is done using the LaTeX typesetting system. The group members can use any text editors they like. A script for compiling the documents to their final format is also available.

8.2.2 Design

In designing the produt some UML and other diagram tools are needed.

8.2.3 Programming

Some C and Java compilers and development environments are used.

8.2.4 Testing

In unit testing JUnit and Cppunit can be used.

8.2.5 Support systems

Other tools used during the whole course of the project are CVS, TWiki web system, E-mail and other communication means.

CASE tool	Communication	Planning	Documentation	Reporting	Modeling	Integration	Version management	Building	Prototyping	Method-support	Language-processing	Program analysis	Testing	Debugging
Email	x													
LaTeX			x	x										
CVS						x	x							
Team Wiki	x			x										
XFig, Dia					x									
C, Java											x			
Make			x					x						
Turjo's progress report script				x										
Gantt project		x												

Table 4: CASE tools selection and distribution

References

1 T.Alanko, Y.Bogoyavlenskiy, *The Plan of Practical Actions for the Developing of the Cooperation between Departments of Computer Science of the Universities of Helsinki and Petrozavodsk.* Release 2.0, 18 of April 2001. Universities of Helsinki and Petrozavodsk.

2 J. Taina, D. Korzun, T. Tuohiniemi, T. Alanko, Y. Bogoyavlenskiy, *Software Engineering Project: Distributed Approach.* Release 1.0, January 2004. Universities of Helsinki and Petrozavodsk.