LIFE Project Number

LIFE02 ENV/EE/000426

PROGRESS REPORT No. 2

Covering the project activities from 01.02.2002 to 31.07.2003

Reporting Date

31/08/2003

LIFE PROJECT NAME

e-System for real Time Democratic Land-Use Planning of Urban Environment – Pilot Action in Narva Municipality (eCommunity)

Data Project

Project location	**Narva**
Project start date:	**01/09/2002**
Project end date:	**31/08/2005**
Total Project duration (in months)	**36 months**
Total budget	**1590589 €**
EC contribution:	**768205 €**
(%) of total costs	**48.30**
(%) of eligible costs	**48.77**

Data Beneficiary

Name Beneficiary	**Narva City Government**
Contact person	**Male, Rauno Schults**
Postal address	**Peetri plats, n° 5, 20308, Narva Estonia**
Visit address	**Peetri plats, n° 5, 20308, Narva Estonia**
Telephone	**+ (372) 3599060**
Fax:	**+ (372) 3599051**
E-mail	**Rauno.schults@narvaplan.ee**
Website	**www.narva.ee**

Table of Contents

2. Lists of (i) key-words and (ii) abbreviations 3

3. Executive Summary 3

4. Project Management 4

5. Technical Development 6

6. Problems encountered 9

7. Dissemination 9

8. Envisioned progress 10

9. Financial issues 11

10. Progress and planned activities (Gantt-chart) 12

ANNEXES

Annex 1 Notification of the assignment of the subcontracts with the partners
Annex 2 Minutes from the PIU meeting – Technology Day 02 – 03 March 2003
Annex 3 Report of the deliverable – Intelligent Agent Software
Annex 4 Screenshots from user interface
Annex 5 System, user and general requirements of the ID- CARD project in Estonia
Annex 6 eCOMMUNTIY Newsletter
Annex 7 Report of the system integration strategies
Annex 8 Minutes from public event Community Day 21 February 2003

2. Lists of (i) key-words and (ii) abbreviations

- e-democracy
- urban environment
- land-use planning
- ICT
- Decision Making Module (DMM)
- online GIS maps
- public involvement
- Intelligent Agent Software

3. Executive Summary

There are several main objectives (listed below) that are kept in mind so that the eCommunity project is going to be a success.
The main objectives of the eCommunity project are:

- Promote sustainable and democratic urban planning by using opportunities offered by information technology and WWW;
- Promote concept of e-democracy by enabling exchange of opinions and information;
- To raise public awareness;
- To create a system as a tool for urban planning.

To achieve the main objectives a strategy was worked out which consists of several deliverables and outputs of the project team. By delivering the outputs on due time it should be possible to keep the time management of the project. The main outputs for the period of 01 February 2003 – 31 July 2003 are as follows:

- preparation and signature of the consortium agreement between the beneficiary and the partners (see Annex 1)
- dissemination of the project (MS2. Community Day, press releases, introduction in radio etc.)
- creation of the project newsletter on the 25th July 2003 (see Annex 6)
- executive committee meeting (MS3. Technology Day)
- software for Intelligent Agent

4. Project management

Project Management team has taken several steps to reduce the impact of bottlenecks. For more efficient work the following committees were organised:

- Advisory Committee
- Executive Committee
- Technical Committee

A better picture from project management structure is presented on the following scheme.

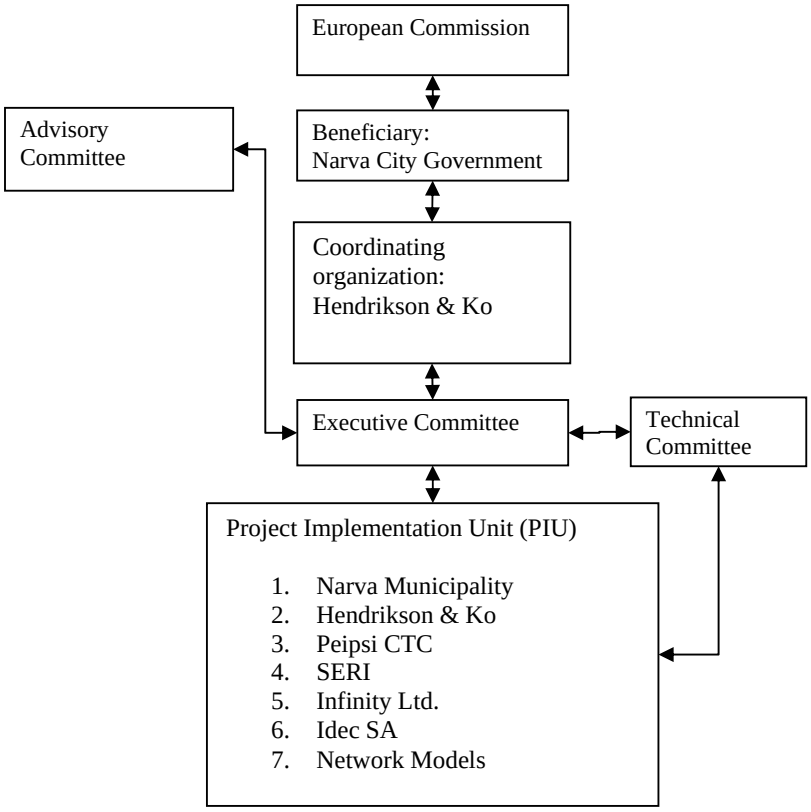


Figure 1 – eCommunity organisation

The contribution of the beneficiary and the partners to the various tasks of the project, as well as the persons in charge of each task, are shown in Table 1.

Table 1 – The role of the beneficiary and partners in the various tasks, and the task leaders. IC = partner in charge (also does most of the practical work), MC = another partner giving a major contribution to a task*****, C = contribution to the task*****.

***Tasks:**

4152 = Management and reporting to the EC
4156 = Creation of user interface
4158 = Creation of process layer and security layer
4159 = Creation of database layer
4160 = Integration of the system
4161 = System launch in Narva community
4162 = Dissemination
4201 = System launch in society level

Tabel 1

	Task 04152	Task 04156	Task 04158	Task 04159	Task 04160	Task 04161	Task 04162	Task 04201
Narva City Government (beneficiary)	MC	MC	C	C	C	IC Mr. Rauno Schults	C	C
Hendrikson & Ko	IC Mr. Vahur Sikaste	C	C	C	C	C	IC Mr. Vahur Siakaste	IC Mr. Vahur Siakaste
SERI	C	MC			C	C	C	MC
Infinity Ltd.	C	MC	MC	MC	IC Mr. Levente Csupor	MC	MC	MC
Network Models R & D Ltd.	C	IC Ms. Eleni Hadjiconstantinou	MC	MC	MC	MC	MC	MC
Idec SA	C	MC	IC Mr. Fotis Givanopoulos	IC Mr. Fotis Givanopoulos	MC	MC	MC	MC
Peipsi CTC	C	C				MC	MC	C

The Project Management team was satisfied with the following key outputs of the project during the reporting period of 01.02.3004 – 31.07.2003:

- dissemination of the project (see Chapter 7)
- executive committee (PIU) meeting (see Chapter 5.2)
- software for Intelligent Agent (see Chapter 5.3)
- creation of the project newsletter (see Annex 6).

The *second* meeting of the project team was called Technology Day and was held in Athens, Greece, on March 02 – 03, 2003. As at the beginning it was planned to hold in April 2003, the project consortium thought it is wiser to organize the meeting earlier as there were some misunderstandings with consortium agreements and in technical issues. On the meeting all the problems were solved.

5. Technical development

5.1. General

The project has continued very successfully, although there have been some problems, which were solved by the PIU as they appeared.

5.2. Management and reporting to the EC (Task 4152)

	Start Date	**End Date**
Proposed	**01/09/2002**	**31/08/2005**
Actual	**01/09/2002**	**31/08/2005**

The task has continued without any major problems, although there were minor delays in the financial reporting part of the Partners. The delays were forced by the reorganization of the Partner companies. After the organizational changes there are new project managers in the following companies:

- Infinity Ltd. - Mr. Levente Csupor
- Idec SA - Mr. Fotis Givanopoulos
- Peipsi CTC - Mr. Erkki Vedder

As the changes took place in summer the new project managers needed some additional time to adopt oneself to eCommunity project.

Nevertheless the task was very successful as all the actions took place as planned. Some activities were organized even earlier (MS3. Technology Day; see Annex 2). Also some tasks (4160 and 4162) were started earlier and that had a very positive impact on the project, as it is helping to finish all the tasks on due time.

Although the second Progress Report is sent with the delay there will be no changes in reporting procedures and the next report is sent as planned on the 24th February 2004.

5.3. Creation of user interface (Task 4156)

	Start Date	End Date
Proposed	**01/09/2002**	**31/08/2003**
Actual	**01/09/2002**	**31/08/2003**

Task 4156 has continued as planned. After the user requirements were described (see previous Progress report) the PIU continued with the developing of the user interface. The main deliverables of the reporting period are:

- The Intelligent Agent Software i.e. IAS (see Annex 3), which is sophisticated software for the user interface module. The IAS is also part of the Decision Making Module and will analyse, by using the artificial intelligent techniques, the feedback that is sent by people to the City Government.
- The user interface (see Annex 4). In Annex 4 are screenshots from the developed user interface. The deliverable will be ready as planned and the responsible Partner (Network Models R & D Ltd.) will make some additional development of the module which is needed from integration point of view.

5.4. Creation of process layer and security layer (Task 4158)

	Start Date	End Date
Proposed	**01/09/2002**	**31/08/2003**
Actual	**01/09/2002**	**31/08/2003**

Task continued without any major difficulties and will be finished as planned. The task was depending very much from the requirements of the Estonian ID – card project and mostly the requirements were in Estonian language, but with good cooperation with Estonian Governmental Information Systems Department (RISO) we managed to translate all the requirements into English (see Annex 5) and so make them usable for all our Partners. The process and security layer, including the Decision Making Module are almost completed and need some testing.

5.5. Creation of database layer (Task 4159)

	Start Date	End Date
Proposed	**01/09/2002**	**28/02/2004**
Actual	**01/09/2002**	**28/02/2004**

Task 4159 progressed as planned by the PIU. No difficulties were faced during the reporting period and the task will end as planned, with the installation of the database at the local level.

5.6. Integration of the system (Task 4160)

	Start Date	End Date
Proposed	**01/06/2003**	**28/02/2004**
Actual	**20/09/2002**	**28/02/2004**

As the task commenced earlier and proceeded without any bigger problems, we see that the task 4160 will end as planned. Although the task was started earlier it will not affect the budget of the task. The earlier start of integration task was enforced by the Partner in charge (Infinity Ltd.) to anticipate possible problems that may come up at the end of the task. This is also a valuable lesson we learned, that to start with some tasks earlier may have a very positive impact on the project as it happened with the Integration task 4160. Already at the start of the project the Infinity Ltd. worked out a strategy for the Integration of all modules (see Annex 7 - Systemplan) and all partners had to respect the guidelines and the limitations of integration then developing their modules.

5.7. System launch in Narva community (Task 4161)

	Start Date	End Date
Proposed	**01/09/2003**	**31/08/2005**
Actual	**01/09/2003**	**31/08/2005**

The task will begin in 01 September 2003 as planned for further developments see Chapter 7.

5.8. Dissemination (Task 4162)

	Start Date	End Date
Proposed	**01/12/2003**	**31/08/2005**
Actual	**01/03/2003**	**31/08/2005**

The task 4162 commenced earlier as there were very many interested stakeholders. Also it is important to mention that the earlier start of the task had a very positive impact on the project as PIU could exchange experiences with other projects around the EU. For more detailed description of the task see Chapter 7.

5.9. System launch in society level (Task 4201)

	Start Date	End Date
Proposed	**01/03/2005**	**31/08/2005**
Actual	**01/03/2005**	**31/08/2005**

The task will begin on 01 March 2005 as planned.

6. Problems encountered

Slight delays occurred in the financial reporting of Partners due to the organisational changes in partner companies, but it does not have any substantial impact on the project as all the important actions and deliverables were completed in due time. Problems are also described in Chapter 4 and 5.2

7. Dissemination

Dissemination is considered as very important task of the eCommunity project and that is the reason why it was started earlier with this task.
The key dissemination and publicity outputs of the reporting period were as follows:

- Community Day (see Chapter 7.1 and also Annex 8)
- further development of the project website www.narvaplan.ee/e-com (see Chapter 7.2)
- dissemination of the eCommunity project on conferences and public events (see Chapter 7.3)
- dissemination in newspapers, advertisements in local radio and television (see Chapter 7.4)
- creation of the project Newsletter

7.1. Community Day (see Annex 8)

Altogether 26 persons participated on the Community Day event. The project team presented the main idea of the eCommunity system and the benefits to local people. Mr. Uuno Vallner (RISO) presented the ideology of connecting the ID – card system with the eCommunity system and explained how to use the ID – card. Also was organized a discussion to get feedback from local stakeholders.

7.2. Further development of the project website www.narvaplan.ee/e-com

The project website was updated and the structure was improved on 27th June 2003.

7.3. Dissemination of the eCommunity project on conferences and public events

The project team has presented the ideology of the eCommunity project on the following events:

- International conference Siena 27th – 28th March 2003: Delivering Sustainable information cities; presentation by Mr. Roman Mesicek (SERI) – eCommunity System for Planning Urban Environments
- LIFE – Environment program seminar on 20 May 2003, Tallinn. On the seminar Mr. Vahur Sikaste shared experience with other participants of the meeting about applying and working with LIFE projects, also valuable lessons learnt during the project.

- Conference “e-Elections: challenge for the community”. Mr. Vahur Sikaste participated on the conference. He distributed information about the project (distribution of the project abstract and webpage address).
- Aarhus Convention Task Force on Electronic Information Tools Meeting, 23 – 24 June 2003, Sofia. Mr. Vahur Sikaste presented the project – “eCommmunity - e-System for Real Time Democratic Land-Use and Planning of Urban Environment „ On the conference was a panel about best practices around the world, which was very informative and lucrative for eCommunity project. The eCommunity project team has already analyzed the information that was on the following web pages:
    - Malta environment and planning system; www.mepa.org.mt
    - Swedish environmental ICT system; www.svenskamiljonatet.se

7.4. Dissemination in newspapers, advertisements in local radio and television

During the reporting period were the following articles and press releases published at local level:

- The project team published several advertisements in local Narva newspaper, television and radio.
- Article in Eesti Postimees – “In Narva commenced eCommunity project.” http://vana.www.postimees.ee/index.html?op=lugu&id=102500&number=817&rubriik=3
- http://www.narvaplan.ee/uudised/NG261102.jpg (in Russian language)
- Article about eCommunity and the project team good experiences – “International Intranet system”; http://www.aripaev.ee/2410/rubr_artiklid_241010.html

7.5. Creation of the project Newsletter

The Newsletter was distributed to following stakeholders:

- Mr. Andrius Plepys, Institute for Environmental Engineering, Lithuania
- Mr. Janis Plavinskis, Latvian Pollution Prevention Center, Latvia
- Ms. Lina Sleinotaite-Budriene, KTU Aplinkos Inzinerijos Instituto, Lithuania

8. Envisioned progress up to 24/02/2004

The key deliverables and outputs the next reporting period, 01 August 2003 – 24 February 2004 are as follows:

- continuous dissemination of the project
- end of task 4156, ends with the deliverable - user interface
- end of task 4158, ends with the deliverable - process and security layer, including DMM on 31 August 2003
- project team meeting (PIU) on the 26th of September in London
- preparation of the guideline materials for the implementation of the project
- preparation of the strategy for public involvement
- Interim report is sent on the 24th February 2004 and is covering the period 01/09/2002 – 31/01/2004

9. Financial issues

PROJECT COSTS INCURRED

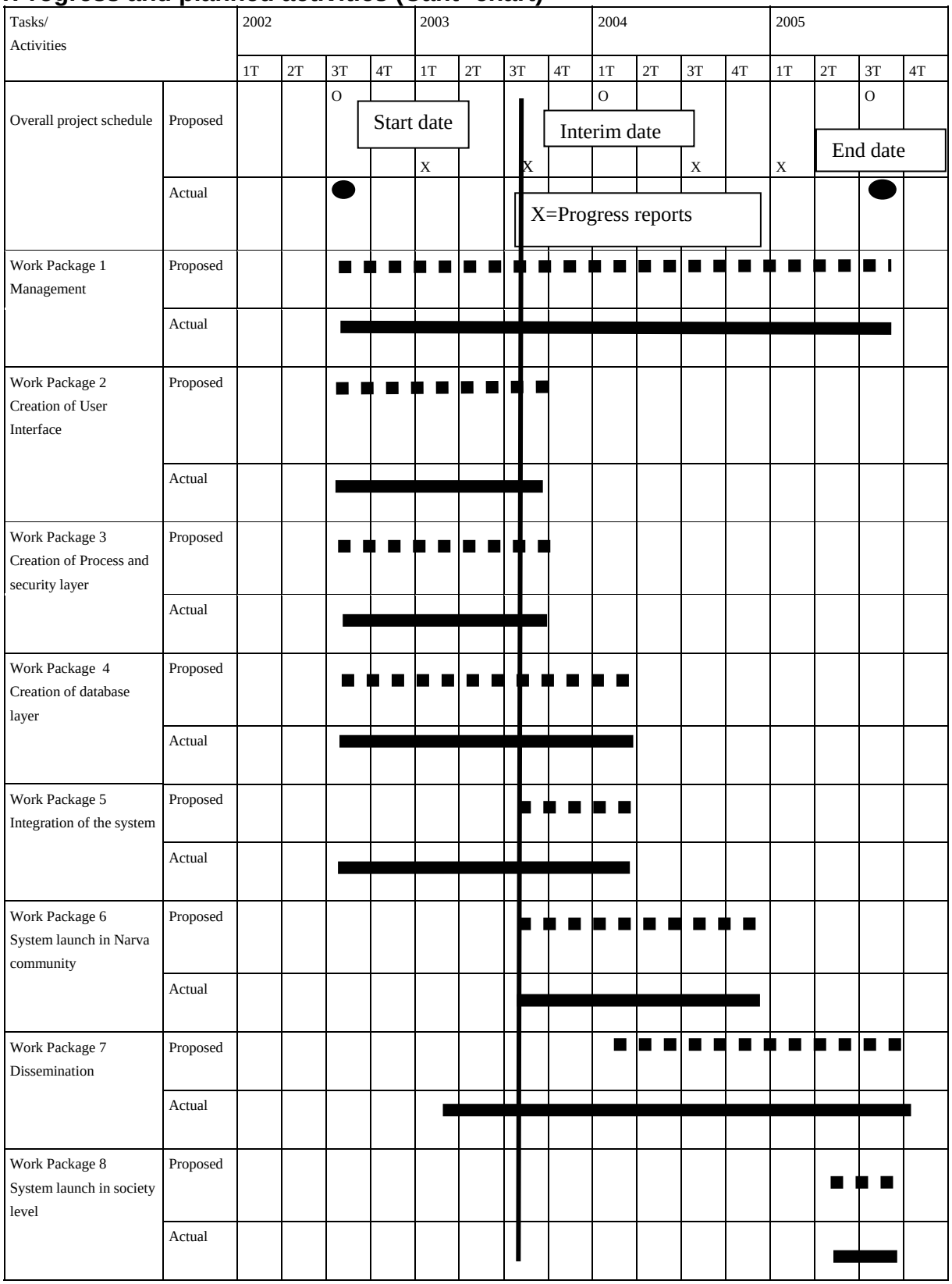
	Cost category	Total cost according to the Commission's decision	Costs incurred from the start date to 31.07.2003	%
1.	Personnel	1154650	312 622,96	27,08
2.	Travel	55700	14 610,71	26,23
3.	Outside assistance	105000	50 416,70	48,02
4.	Durables: total non-depreciated cost	30800	17 255,62	56,02
	- Infrastructure sub-tot.		-	
	- Equipment sub-tot.	30800	17 255,62	56,02
	- Prototypes sub-tot.		-	
5.	Consumables	101400	8 985,70	8,86
6.	Other costs	40000	19 181,78	47,95
7.	Overheads	103039	44 352,79	43,04
	SUM TOTAL	**1 590 589**	**467 426,27**	29,39

Comments on the budget costs:

4. Almost 50 % of the project equipment costs have occurred until 31 July 2003, as there is bought several computers for project team.
5. Most of the consumables cost will occur later (book, video, CD-ROM etc.)
6. Almost 50 % of the foreseen costs have occurred, as planned, as it was needed at the beginning of the project.

The 40% threshold of eligible costs is expected to be reached in December2003 - January 2004. The interim report will be submitted by 24 February 2004, as planned.

10. Progress and planned activities (Cant- chart)



ANNEXES

ANNEX 1

Referring to Article 4.7 of the LIFE standard administrative provisions, the Finnish The Narva City Government hereby notifies the Commission on the subcontracts that Narva City Government, as a beneficiary, has concluded with the partners of the eCOMMUNITY project (LIFE02 ENV/EE/000426).

The principal contents of the agreements are as follows:

Parties of the agreement are defined at the beginning of the subcontract

Paragraph 1 specifies the scope and purpose of the agreement. A reference is made to Appendix 1 that consists of the revised eCOMMUNITY proposal, accepted by the Commission. The paragraph stipulates that the roles, responsibilities and tasks of the parties are in detail described in the accepted proposal. The paragraph also stipulates that the Community may excercise the same rights and guarantees vis-à-vis the partners as the beneficiary himself.

Paragraph 2 specifies the duration of the project.

Paragraph 3 stipulates the financial arrangements between the beneficiary and the partners. A reference is made to Annex 1.

Paragraph 4 stipulates rules for technical and financial reporting, describes the partner´s part and responsibilities in the project. Also are described the financial procedures of the payments.

Paragraph 5 specifies the contact addresses for the communication.

Paragraph 6 stipulates the Annexes of the subcontract. The paragraph also stipulates that the Community may excercise the same rights and guarantees vis-à-vis the partners as the beneficiary himself. Also is specified that the partner has to act according to the rules in Standard Administrative Provisions (SAP)

The Annexes attached to the agreement (cf. Par. 6) are as follows:

1. The revised eCOMMUNITY proposal, accepted by the Commission
2. Standard Administrative Provisions (SAP)

ANNEX 2

eCommunity workshop „Technology Day“
02/03 – 03/03/2003
Idec SA office, Piraeus, Greece

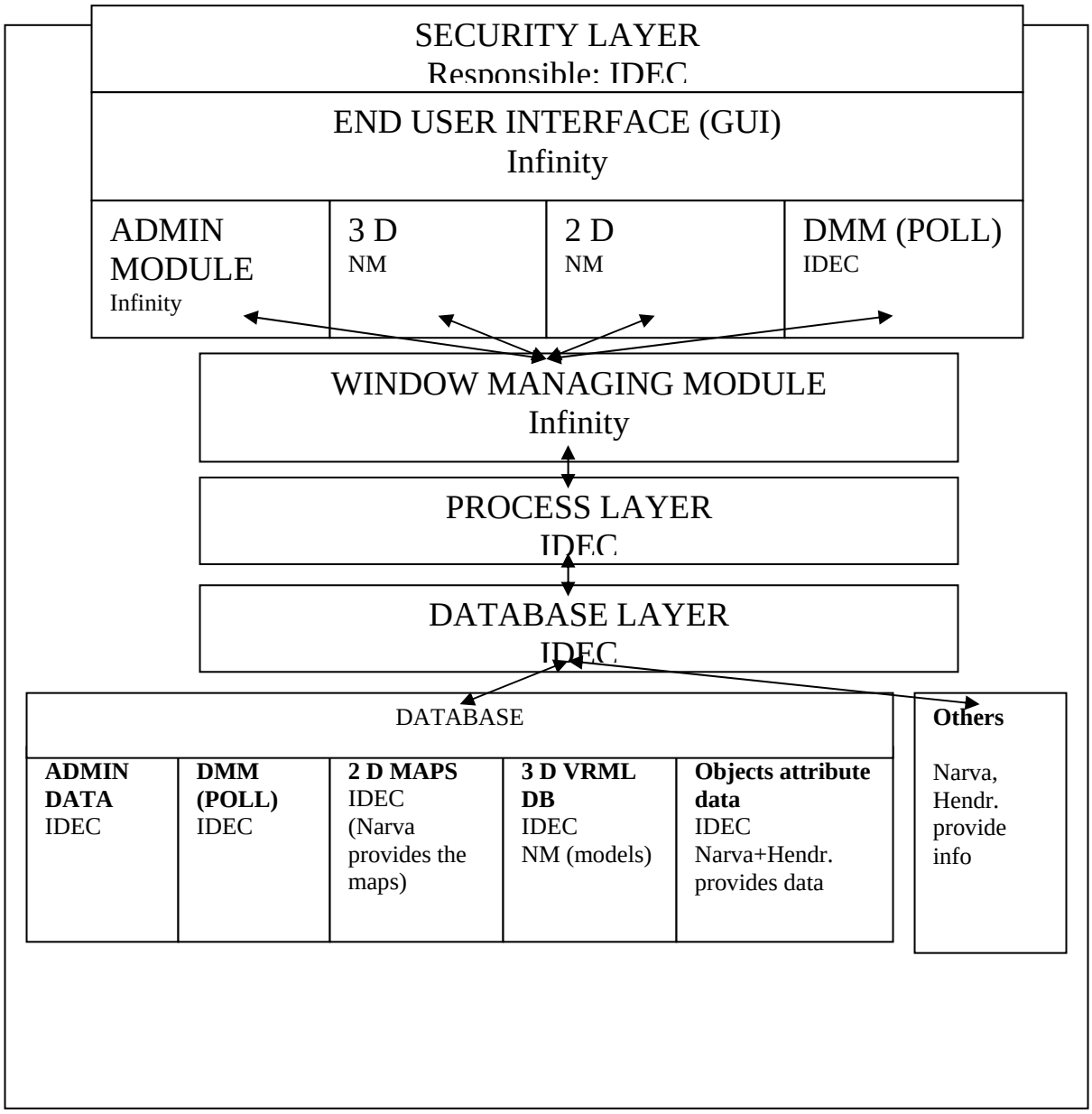
Participants:

1. Tanel Mazur – Narva Municipality
2. Rauno Schults - Narva Municipality
3. Panos Katsambanis – IDEC
4. Elena Manolakaki - IDEC
5. Dimitris Kamenopoulus - IDEC
6. Peep Leppik - Hendrikson & Ko
7. Vahur Sikaste - Hendriksno & Ko
8. Doris Schnepf - SERI
9. Eleni Hadjiconstantinou - Network Models
10. Varnavas Serghides - Network Models
11. Nicos Christofides - Network Models
12. Beata Raboczki - Infinity Ltd.
13. Geza Lucz - Infinity Ltd.

DAY 1: 02. 03. 2003.

1. Opening the Technology Day
2. Summary of completed inputs and internal deliverables. (Tanel Mazur)
3. General vision and theoretical needs of the system. Presentation by Peep Leppik.
4. System Integration plan. Presentation by Geza Lucz.
5. Creating the 3D and 2D module for the system – functionality, requirements, technical specifications, available technologies and limitations of the software's. Presentation by Varnavas Serghides. Summary of the presentation.
   - System limitations:
     → We have limited time resources
     → Detail of the modules is limited
   - The system id depending from several general requirements:
     → Flow rate
     → 2 D map of the system is needed to start the modeling process.
     → Objects are generic modules
     → Navigation is provided via standard VRML plugin
     → Navigation limited to a walk – through capability in the planned area
   1. System interactivity to the users
      → Links to information document provided
      → Users are making suggestions through a window what is outside the VRML window
      → Links between the documents and button on the 2 D page
      → Displayed features with a starting point in 3 D
6. The development of database, security and process layer. Presentation by Dimitris Kamenopoulos
7. Presentation about available data (Tanel Mazur).
   Questions:
   2. What data is available from buildings in digital form.
   3. Number and description of different models.

8. System development plan



9. Tasks of the partners.

No	Description of the tasks	Responsible	Deadline
SECURITY LAYER			
1	ID card certification specifications	H & Ko	19/03/2003
2	Data for X-road i.e. additional specifications	H & Ko	19/03/2003
3	Description what each user level will provide	Narva	19/03/2003
END USER INTERFACE (GUI)			
1	Guidelines, how should the end user interface look.	Narva, H & Ko, SERI	05/04/2003
3 D AND 2 D MODULE			
1	MAPS (Vector maps)	Narva	05/04/2003 – until the end of the year.
2	Generic building data (shape & size)	Narva	05/04/2003
3	Textures (bitmap)	Narva	05/04/2003
4	Coordinates	Narva	05/04/2003
DMM (POLL)			
1	Examples of usually used polls and questionnaires	H & Ko	5/05/2003
WINDOW MANAGER			
1	Service requirements of 3 D, 2 D and DMM modules	IDEC, NM	Needs to be specified
PROCESS LAYER			
1	HTML interfaces of everything	NM., Infinity	Needs to be specified
DATABASE LAYER			
1	Outputs of database	Idec	Needs to be specified
ADMIN DATABASE			
-	-	-	-
ADMIN MODULE			
1	Requirements for databases	Idec, NM, Infinity	Needs to be specified
2	Requirements of DMM, 2D, 3D	Idec, NM	Needs to be specified

10. Description of the possible platform for the system
- Open source operating system
- One server
- My SQL Database
- php (Java and Servlet ???)

11. Needed inputs from PIU
    - Data for terrain (Responsible: Narva)
    - Investigation for 2D Graph Editing Plugin (Responsible: NM)

DAY 2: 03. 03. 2003.

1. Summary from previous day. General points of the development plan of eCommunity system tool:
   - Fixed platform – the fundament on what to create the system
   - All partners know their tasks and there is a common understanding
   - We have set a more concrete timetable for sub-tasks
   - There is a common understanding in system and user requirements, technical specifications and in technical limitations.

2. There are set some communication requirements for all partners of the PIU:
   - For e-mails a partner has to answer within 48 h
   - On the intranet will be set up a reporting plan for financial issues, and issues concerning overall reporting to the beneficiary and EC. This is done to avoid unnecessary expenditures and time loss.

3. At the second part of the day were solved all opened questions concerning contracts and financing.

ANNEX 3

Report on Intelligent Agent Software

LIFE-Environment

e-System for Real Time Democratic Land-Use Planning of Urban Environment – Pilot Action in Narva Community (eCommunity)

LIFE02 ENV/EE/000426

Deliverable (Task 1.1)

Report on the Intelligent Agent Software for the Automated Extraction of Information

Dr E. Hadjiconstantinou
Dr V. Serghides
Mr. E. Rappos

Network Models
London

May 2003

Table of contents

1 Introduction 3
2 Software Overview 3
3 Design characteristics 5
3.1 User input files 5
3.2 Initialisation 6
3.3 Preprocessing 6
3.4 Part of Speech tagging 7
3.5 Grammatical Training 7
3.6 Grammatical Tagging 9
3.7 Unknown words 10
3.8 Numerical underflow 10
4 Analysis and Output 11
5 The ODBC option 12
APPENDIX A: 15
A.1 Program Execution: diagrammatic representation 15
A.2 Brief User Guide 15
APPENDIX B: System and user-defined input files 18
B.1 Configuration file 18
B.2 Tag set file 19
B.3 Dictionary (lexicon) files 20
B.4 Substitution file 20
B.5 Synonym file 21
B.6 Ignored verbs file 21
B.7 Pattern file 22
B.8 System parameter settings 22
APPENDIX C: Grammatical tag set 23
APPENDIX D: An Illustrative Example 23
REFERENCES 28

1 Introduction

This report describes the development of an Intelligent Agent software for the automated extraction of key information from a collection of user responses on a given topic.

The main part of the report focuses on the way this automation is performed and the theoretical background (relating to artificial intelligence) that is necessary to understand the process. Many technical details regarding the implementation of the procedure are also described.

The report is organised as follows: Section 2 presents an overview of the software and Sections 3 and 4 contain a detailed description of the Intelligent Agent. In Section 5 the option of database connectivity and the associated relevant characteristics are discussed. Appendix A shows a diagrammatic representation of the program and a short program user guide. Appendices B and C provide details on the various system and user-defined input files specified in the software. Finally an illustrative example is described in Appendix D.

2 Software Overview

The software takes as input a collection of user replies (messages) and by using artificial intelligent techniques, extracts a set of sentences which have been identified to be of high significance and presents them as an output. Various levels of significance for a given statement can be determined based on user-defined parameters such as frequency of message or particular grammatical structure required. These parameters affect the level of detail presented in the output, for example, one may be interested only in a very short list of the most important points to be output, instead of a more detailed and longer list of messages.

The software is implemented in C++ and is configured using a special configuration file in which the user-defined parameters can be changed. A detailed description of the format of this file and the meaning of each parameter are given in Appendix B.

The software has the advantage of being connected with a database implemented using ODBC (Open Database Connectivity). This means that the input of the program can be retrieved directly from a database, e.g., the database where the messages were originally stored after a user has entered his response through the internet. Similarly, the output of the software can be stored directly in a database, if required. The option to use a database is

specified in the configuration file; alternatively the user can choose to use text files instead. ODBC provides database connectivity for all contemporary types of database such as Access, Excel, Dbase, FoxPro, Paradox, Oracle or SQL server. The same program can run with any of these data sources without any modification.

The software is structured as follows:

Input

- Configuration and other input files (text files)
- A set of text messages (from a database or text files)

Output

- A summary of text extracted from the set of user messages representing statements of high significance (grammatical or verbal) determined by the Intelligent Agent.

Process

1 *Text Initialisation and processing:* The messages are read and processed, in order to make the text suitable for the next steps.

2 *Grammatical tagging*: Each word of a message is analysed grammatically and the relevant grammatical attribute is identified. The principle behind this is that many users may express the same view/opinion in different ways. This step aims to filter out irrelevant information and only focus on the important parts of a message. For example, two messages from different users may contain the following sentences:

Message 1: "*I would like to build the school.*"
Message 2: "*It will be nice to build the school.*"

By studying the grammatical structure of the statements, the Agent can decide that 'build the school' is the important part of the statements whereas everything else is just 'flavour' specific to the person's style of writing or speaking.

3 *Grammatical analysis*: The software automatically searches for grammatical patterns common to many messages. Once identified, the corresponding sentences are processed.

4 *Statistical analysis and output*: For the statements identified above, a complete statistical analysis is performed to determine their relative significance. Finally, the sentences with high scores are output.

It must be highlighted that the Intelligent Agent models and algorithms have been designed for any application of user text messages irrespective of the language used. However the dictionary file and the set of grammatical tags (see Appendix C) supplied with this version of the Intelligent Agent software are for the English language.

3 Design characteristics

This section discusses in greater detail various software aspects as well as the models and algorithms used in the development phases of the Intelligent Agent. A diagrammatic representation of the program execution is given in Appendix A.

3.1 User input files

The program takes input from the user in the form of two text files. These are called **the training corpus** and the **main corpus**. If the program is used in ODBC mode (with database connectivity) these files are created from the messages of the database. The main corpus contains all the text that will be analysed (complete set of messages). The training corpus is used in the *training* process of the system, and it is described in detail in Section 3.5. Training is a necessary requirement for the grammatical analysis since it aims to study the grammatical connections between the words in order to perform the grammatical tagging later on. The procedure requires that a sample text (the training corpus) is processed. The training corpus can either be the same as the main corpus or, more commonly, a subset of it. This is because the training process is complex and time consuming, so using an extremely large training corpus will result in large computational time requirements. If using ODBC, the user defines N to be the number of messages that would be desirable to use in the training process, then the first N messages are read and processed for training.

3.2 Initialisation

This is the initialisation part of the algorithm. The configuration file is read and processed and various internal data structures are initialised. Then the program reads the global dictionary (lexicon) file, the tag set file and the substitution file, which are necessary for its operation. Then the optional user dictionary, user patterns and synonym files are processed and some checks are performed for consistency. If the user specified to use database connectivity (ODBC) then the specified database is opened and the list of messages extracted to a temporary main corpus file. A specified subset of the messages is also extracted to a temporary training file as described above. If ODBC is not used, the user-specified main and training files are read.

3.3 Preprocessing

Preprocessing is performed on the training and main corpus before the algorithm begins. The purpose is to ensure that these files are of the required format to be analysed. Preprocessing is performed on both files and in three stages:

Stage (i) - Substitution: A list of substitutions is provided. This stage aims to replace abbreviations with complete words for more clarity. The list of substitutions is described in the "substitution file", which is a text file given as input. Examples of such substitutions are shown in Table 1 and more details are provided in Appendix B.

Word	*substitution*
n't	not
I'm	I am
won't	will not
'll	will
etc	

Table 1: An example of possible substitutions

If a substitution requires the addition of a whitespace at the beginning of the replacement string, the special character "~" can be used as a distinct marker to separate the resulting string from preceding words. Instances of "~" will be converted to spaces in the second stage of preprocessing. For example, the word "I can't" will be substituted by the string "I can~not".

Stage (ii) - Punctuation: In the context of this application punctuation marks (commas, full-stops, etc.) and symbols (“#”, “-” etc) are separated from other words and they are treated as one-letter ‘words’ , belonging to special grammatical categories (see Appendix C). These symbols are:

```
.,:;!?()[]{}-_"#/@'$\£&€
```

Finally any instances of the special character “~” are replaced by whitespace.

Stage (iii) – Unknown words: The third part of preprocessing splits the two corpus files into sentences and checks for unknown words (words not in the global or the user dictionaries). If such words are found the program continues assuming that each word can take any possible tag. They are saved to a text file, whose name is specified by the user in the configuration file, so that the user can add these words to the user dictionary and improve the reliability of the analysis.

3.4 Part of Speech tagging

This step assigns tags to each word of the text referring to the part of speech they belong. Such methods [1–6] are commonly used in areas of Artificial Intelligence where content analysis is required.

There are two approaches to part of speech tagging: supervised and unsupervised. For supervised models, an already tagged corpus is required, whereas in unsupervised models, such as the one implemented here, a training corpus is used to enable speech tagging. The latter approach is preferred for two reasons: (i) tagged corpuses are context specific, (for instance a tagged corpus containing poetry is not usable for areas outside that field) and (ii) an unsupervised method uses a training corpus *derived from* the application for which it will be used and so provides more accurate results. On the other hand, the disadvantage is that training can be a time-consuming process, however this is not a serious limitation for the Intelligent Agent since real-time results are not required.

3.5 Grammatical Training

The model used by both the training and the tagging procedures is the *Hidden Markov Model.*

In this setup, each tag assigned to each word of the text to be tagged (i.e., the output of the tagging algorithm) is a random variable $X_1, X_2, \ldots, X_T$ where T is the total number of observations (words) in the corpus.
The possible part of speech tags $t_1, t_2, \ldots, t_n$ represent the states $s_1, s_2, \ldots, s_n$ of a Markov chain where the transition probabilities (moving from a word of tag i to a word of tag j in a sentence) a_{ij} are given by

$$a_{ij} = P(X_{t+1} = t_j \mid X_t = t_i),$$

where $t = 1, \ldots, T$ and the initial state probabilities are given by

$$\pi_i = P(X_1 = t_i).$$

Additionally, at each transition between the tags of the Markov Model there is a probability that a given word w_k will be *emitted* is given by

$$b_{ij}^k = P(O_t = w_k \mid X_t = s_i, X_{t+1} = s_j)$$

These probabilities depend on both the states where the transition takes place, as well as the emitted word. The emitted words are modelled as random variables, O_t ('observations'), which represent the observed output of the Markov Model that we want to match to the sequence of words in the tagging text.

In part of speech tagging however, the emission probabilities b_{ij}^k do not depend on the destination state, so for simplicity we shall use

$$b_i^k = P(O_t = w_k \mid X_t = s_i).$$

Note that in the above formulae the probabilities do not depend on t (i.e., the particular position of the word in the corpus) but rather on the relationship between the states. This behaviour (stationarity) is a characteristic of the Markov model.

The way that the model works is the following:

Hidden Markov Model speak:

> *"Given a set of states (with given transition probabilities), the model moves from one state to another T times. Each time a transition is made, a symbol (from a given symbol set) is emitted (with a probability that depends on the states and the symbol). We can only observe the T symbols that were emitted, but we can't observe the sequence transitions. How can we find the most likely sequence of state transitions that could have emitted those symbols? "*

Part of speech tagging speak:

> *"The states are the possible grammatical tags (e.g., verb, noun, pronoun, etc) the model moves from state to state for a time period T, in a secret fashion that we cannot observe. For each transition, (e.g., from 'verb' to 'noun') there is a probability that a particular word from a dictionary will be emitted. Given that we know the words that are emitted, which is exactly the sentence that we want to tag, how do we determine what is the most likely sequence of tags (grammatical categories) that could have generated these observations? "*

The implementation of the above is performed using the Baum-Welch algorithm and the forward-backward procedure. This procedure reads the training corpus and calculates the required probabilities that are necessary for the part of speech tagging. See [4] for more details.

A further problem is parameter estimation in the Baum-Welch (forward-backward algorithm), that is, to estimate the values of *a* and *b* from the training data. This can be done with standard re-estimation techniques using the training corpus, e.g. [4].

In may be noted that unsupervised models also require a set of initial values for the probabilities, i.e., the values they have before the training starts. During the training process these probabilities will be re-estimated to reflect the training corpus. We use the Jelinek's initialisation method, which is described in [4].

3.6 Grammatical Tagging

Tagging refers to the assignment of part of speech tags to each word of the corpus. If the values of

$$\Lambda = \{a_{ij}, b_j^k, \pi_i\}$$

are known, then the tagging is performed as follows: We essentially seek the sequence of states (grammatical tags) that best explains the observations (sentences).
This means that, for each t, we would find the X_t that maximizes

$$P(X_t \mid O, \Lambda).$$

The 'best' value of X_t is denoted $\hat{X}_t$.

$$\hat{X}_t = \arg\max_X P(X \mid O, \Lambda).$$

The above maximisation is performed with the Viterbi algorithm [4,5], which produces the 'best' (most probable) sequence of grammatical tags corresponding to the given text.

3.7 Unknown words

Unknown words can be handled by assuming all possible tags for each unknown word. If the training corpus contains many instances of the unknown word, then the algorithm can correctly derive the part of speech of the word from the way it is used in a sentence. The program reports all unknown words encountered and saves them in the relevant file.

3.8 Numerical underflow

Both the Viterbi and the Baum-Welch algorithms require multiplications of very small probabilities.
For instance, the b's represent the probability that a given word will be ejected during a transition from a possible tag. If the tag is, say, NN (noun) and our dictionary contains 10000 nouns, then the b's will be of the order of 0.0001; multiplying a few such values together will cause numerical underflow and the computer will not differentiate the resulting number from zero. Moreover when dividing two numbers that are very close to zero, any small change in one of the numbers will be amplified in the output.

We have implemented two different procedures in order to deal with the issue of numerical underflow.

(i) The Viterbi algorithm involves only multiplications and divisions of small probabilities. It is, therefore, possible to use the *logarithms* of the probabilities throughout, replacing multiplications with additions. This approach is implemented in this algorithm.

(ii) The Baum-Welch algorithm requires a different approach since the algorithm involves additions as well as multiplications. In each step of the algorithm divisions of small numbers are necessary. In this case, we calculate a scaling coefficient with which the numerator and the denominator are multiplied as explained in [5]. In the calculation of the probabilities these coefficients cancel out.

4 Analysis and Output

Once each word in the corpus is tagged with the corresponding part of speech, the analysis of the input is performed. First of all the text is read and a set of *patterns* is determined. These patterns are sequences of grammatical tags varying from size 3 to size 8. There are two types of patterns: **user patterns** and **automatically generated patterns**. User patterns are specified in the (optional) pattern file, in case the user wants to search the text for a particular sequence of grammatical categories. Automatic patterns are determined by the Intelligent Agent based on some internally-defined "intelligent" rules and each one is given a score. Then all patterns having a score less than the one specified by the parameter PATTERN_HITS_REQUIRED are ignored. The remaining patterns are considered to be grammatically important. These patterns (and any patterns explicitly specified in the user pattern file) are used to identify key sentences. A score is awarded to each sentence reflecting the number of times each sentence is present in the corpus. Finally the sentences with a score which is at least equal to the value of the parameter SENTENCE_HITS_REQUIRED are presented as the output.

The exact value of these parameters depends on the level of detail in the output required by the user, (for example, is the user interested in finding the 10 or the 1000 most important statements in the messages?) and also the number of text messages. If there is a large number of messages then raising the number of hits required for a statement to be considered important increases the probability that only relevant information is retained. The exact values can only be determined by experimentation as they will depend on the actual number of messages contained and their quality of expression (the level of similarities between them).

The following options may significantly affect the quality of the output.

(i): The **'ignored verbs file'** is a text file containing verbs that should not be included in the output. By construction, every statement of the output must contain a verb, but if the verb is in the aforementioned file then statements including that verb will not be presented. For example, it is useful to eliminate sentences where the main verb is 'like to' or similar verbs, since in such cases the main information is expressed by another verb which follows.

(ii): The **'synonym file'** contains words of similar or equivalent meaning and as such, should be treated the same for the purpose of this analysis. Sentences that differ only in synonym words are considered to be identical and are merged into one statement in the output. For example, a user may want to specify that the words 'truck' and 'lorry' are synonyms, and so are

the words ‘street’ and ‘road’. In this case the statements ‘the truck is on the road’ and ‘the lorry is on the street’ will be considered identical and their scores will be added together.

(iii) The **parameter SENTENCE_DIFFS_ALLOWED** specifies the number of differences (number of word mismatches) that two sentences may have and still be considered identical. The default value is 0 (no differences tolerated) but experimentation has shown that values of 1 or 2 sometimes give better results as they allow merging statements that are identical in meaning but stated in a slightly differently way. This option can be thought of as an automated version of a synonym file. Note that a value of more than 3 is not recommended as it could possibly distort the output.

5 The ODBC option

This section presents useful information for a user who wishes to use a database in order to retrieve the messages and store the output.
ODBC is a unified method with which a database can be accessed in Windows. This means that a program that needs to use a database can do so using standardized SQL queries, without the need to know the name or type of database to be used. This is automatically performed by the ODBC drivers.

As an example, consider that an Access database is used.

Input table:
The input data is stored in a table called ‘input’ in the Access file. This table includes two columns namely the column containing the message (referred to as ‘text’) and a column called “attitude” which contains the numbers 0, 1 or 2 used to classify a message. This column aims to classify the output into three tables, the ‘positive’ (value 0) table containing the positive statements, the ‘negative’ (value 2) table containing the negative, and the ‘neutral’ (value 1). All these names are configured by the user in the configuration file.

Output table(s):
The output table(s) are required to have the following structure:

Columns for Tables ‘positive’ and ‘negative’:

frequency, messageID, pattern, votes, length, word1, word2, … , word11, word12.

Columns for Table 'neutral':

frequency, messageID, pattern, positive, neutral, negative, length, word1,…, word12.

If the list of attitudes is not present, the output is inserted into the table 'neutral', and therefore only this table needs to be present.

The names of the input and output tables, the input fields (columns) and the database name are defined in the configuration file. An important setting is the parameters ODBCINPUTDNS and ODBCOUTPUTDNS. These denote the DNS name of the database that will be used. These names are defined in the Windows control panel under 'ODBC data sources'. It is necessary for a user to set there a DNS name and a database file name. Then the DNS name is given to the software, the actual file name can be anything and it is found automatically by the ODBC drivers.

The meaning of the output columns is presented below.

'frequency' contains the number of times that a specific statement or point is present in the messages. The sum of all the frequencies may be larger than the total number of messages since one message may contain many different statements.

'messageID' contains the list of ID's of each message, this is assumed to be 1 for the first message of the input, 2 for the second etc. This column allows a user to identify the message (or list of messages) which correspond to a specific output statement and read that particular message(s) in more detail. The inclusion of all the message ID's for thousands of messages however, is not practical as it increases the processing time and also the size of the database. As a result, this field is limited to 200 characters and only the message ID's that fit within this field will be output. This will provide enough flexibility for the user to go back and retrieve a few messages in detail if he/she so wishes.

'pattern' column contains the specific grammatical pattern ID of that statement. Although this number in itself in not very meaningful, it can be used to sort the statements in *grammatically similar* statements, which can help group together related messages.

'votes' represents the number of messages that contain the particular statement. In the 'neutral' table, this column is replaced by three columns, named 'positive, 'neutral', 'negative', denoting the number of messages of each 'attitude' containing that statement.

'length' specifies the length of the underlying grammatical structure that generated a statement message. This is an integer number between 3 and 8. In some cases however, the actual size of the statement may be larger if the Intelligent Agent has determined that a few more words are necessary to obtain a grammatically meaningful statement.

'***word1 – word12***' contain the actual words of the output statement.

If a statement is present only in messages of 'positive' attitude, then it will be output in the 'positive' table and the number of times it was encountered will be stored in the column called 'votes'.
Similarly, if a statement is present only in 'negative' messages, then it will be output in the 'negative' table.
All statements that include 'positive' and 'negative' attributes are output in the 'neutral' table under the corresponding columns.

Note that the output tables are always initialised (empty) each time the program is run. On the other hand, the input table is never altered.

If the ODBC option is not activated, the user must make two text files, one containing the training text and one containing the main corpus (the set of text messages). Running the program will produce an output file called 'output.txt', which will contain the output of the analysis (a list of statements and their frequencies). However, this mode is not recommended because it has limited functionality; in particular, no 'messageID's can be specified and no message 'votes' can be used.

APPENDIX A:

A.1 Program Execution: diagrammatic representation

The main steps of the program execution described in sections 2 – 5 of this report are summarised in the diagram of Figure 1. The number in each box refers to the Section/Appendix where each feature is described.

A.2 Brief User Guide

This Appendix presents a brief user guide that will help you to get started with the software. We assume that a database of messages exists, or the example database 'messages.mdb' could be used. The program is configured by a configuration file where all the user-modifiable options can be changed. The software comes with a default configuration file called 'config.txt', which is a plain text file. A complete description of all the parameters that can be specified by the user is given in section B.1 of Appendix B.

If the ODBC option is used the database to the ODBC drivers and assign a name to the database. This is done in the Windows control panel, under ODBC data sources. The user should choose 'add', then Microsoft Access driver (or a suitable choice if another type of database is chosen) and then select the database file and give it a DSN name, say 'messages'. This is the name that the database is known to the software, and this name can be changed via the configuration file. In this file, the names of the table of the database and the columns (fields) that contain the messages can be specified, but the example database uses the 'default' names so no changes in the configuration file are necessary. There are also some files that must exist in order for the software to run, namely, the global dictionary file (lexicon.txt), the substitution file (substitution.txt) and the tag set file (tagset.txt). These files are supplied with this software and must be located in the same folder as the executable file. To execute the program, the user must type in an ms-dos window:

```
IntAgent config.txt
```

and press return. Once the program terminates, the output is stored in the 'output' tables of the database. Note that the output tables (whose names can also be configured in the configuration file) must exist, otherwise the results will not be stored (an output message will be reported by the program to this effect).

There are many options and parameters that affect the quality and the level of detail presented in the output, which can be modified as required. These are presented in Section 4 and Appendix B.

There is also an option to use text files instead of databases, by setting the appropriate user options accordingly. Information regarding this option is described in Section 4.

Summary of instructions

1. Create database and attach it to the ODBC drivers.

2. Specify user input files (‘dictionary’, ‘synonym’, ‘pattern’, ‘ignored-verbs’ and ‘synonym’)

3. Specify desired parameters in the configuration file.

4. Run the program and check the screen output.

5. Read the output.

6. If an error message is reported, the on-screen message will present instructions on how to proceed,
   or
   if greater level of detail in the output is required, adjust the output parameters, and run the program again.

The Intelligent Agent has been tested using an illustrative example presented in Appendix D. Shown in that Appendix are the sample message database and the output obtained from it.

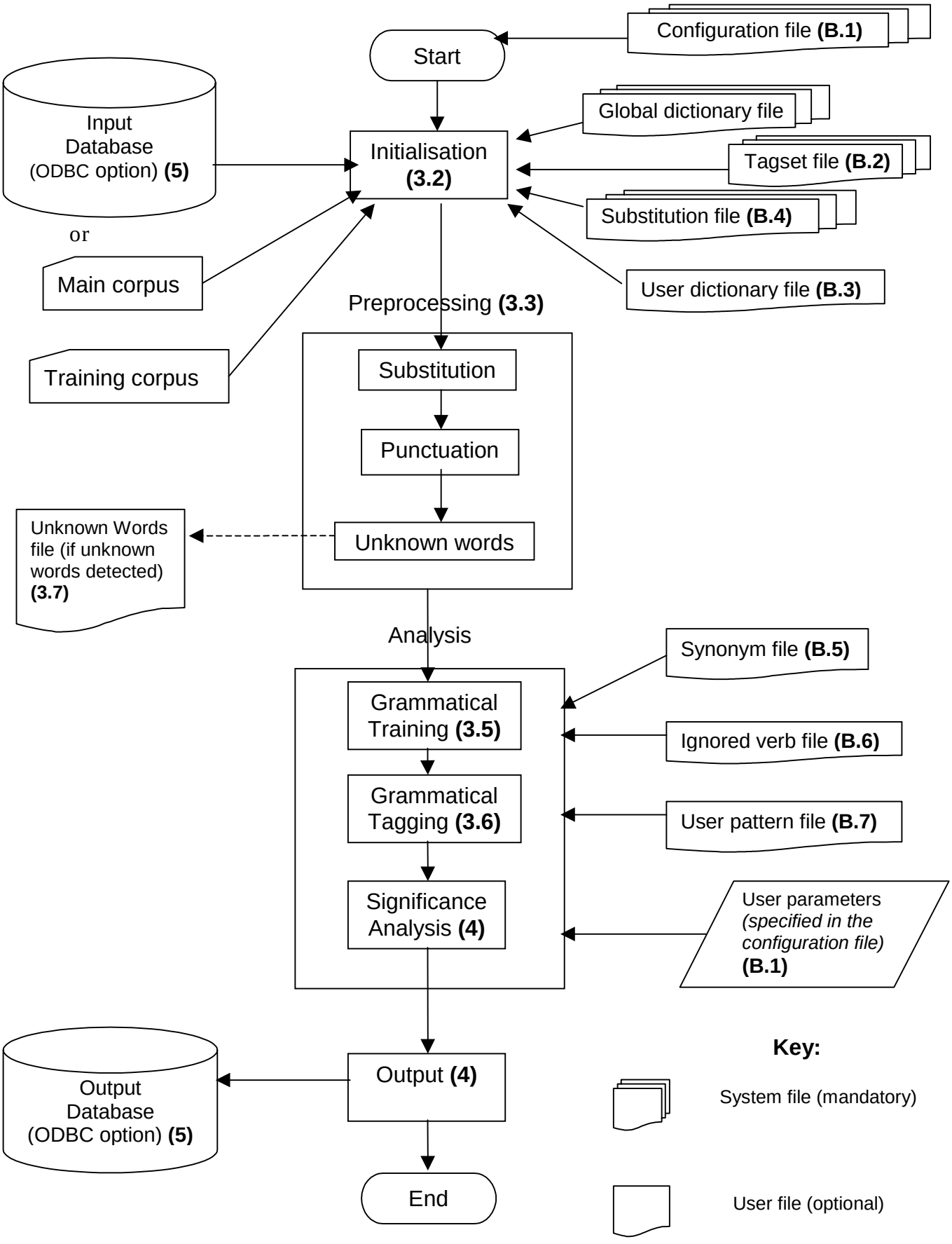


Figure A.1: Program execution outline

APPENDIX B: System and user-defined input files

B.1 Configuration file

The configuration file is a plain ASCII text file, where the values of the user parameters are specified. Each line starts with the name of the parameter, followed by a word (or number) denoting its value. Empty lines are ignored, as well as lines which have the character # at the beginning of the line. This character can be used to add comments.

If a parameter is defined more than once, the last definition will be retained. If you do not define some parameter, a default value is used. If you specify parameters that are not required, e.g., TRAIN_FILE when using ODBC then they will be ignored.

The set of all the parameters is presented below.

Parameter	*Description*	*Default value*
System Input file parameters		
TAG_FILE	Name of the tag set file.	tagset.txt
DICT_FILE	Name of the dictionary file.	lexicon.txt
SUBSTITUTION_FILE	Name of the substitution file	substitution.txt
User (Optional) Input file parameters		
USERDICT_FILE	The user dictionary file.	userlexicon.txt
SYNONYM_FILE	A list of synonym words considered identical in the analysis.	synonym.txt
IGNOREDVERB_FILE	A list of verbs that should be ignored in the analysis. Any statements containing these words will not be output.	ignored-verbs.txt
USERPATTERN_FILE	A list of user patterns to be appended to the automatically generated grammatical patterns before the analysis.	patterns.txt
ODBC parameters		
USEODBC	Specifies whether the messages are read from a database or from text files.	ON
Options if using ODBC		
ODBCINPUTDNS	The DNS name of the database containing the messages.	messages
ODBCINPUTDNSTABLE	The name of the Table of the database containing the messages.	input
ODBCINPUTDNSFIELD_TEXT	The name of the Field (column) containing the message text.	text

ODBCINPUTDNSFIELD_VOTES	The name of the Field (column) containing the message characterization (optional): 0 for positive attitude (agree), 1 for neutral (indifferent) and 2 for negative (disagree).	attitude
ODBCOUTPUTDNS	The DNS name of the database where the output will be stored.	messages
ODBCOUTPUTDNSTABLEPOS	The name of the Database Table with positive statements.	output-positive
ODBCOUTPUTDNSTABLENEG	The name of the Database Table with negative statements.	output-negative
ODBCOUTPUTDNSTABLENEU	The name of the Database Table with neutral statements.	output-neutral
ODBC_NUM_MESSAGES_TRAIN	The number of messages to be used in the training corpus.	1000
Options if not using ODBC		
TRAINING_FILE	Name of the training corpus.	train.txt
CORPUS_FILE	The file containing the text to be analysed (main corpus).	corpus.txt
Analysis options		
PATTERN_HITS_REQUIRED	The number of hits required for a pattern to be considered grammatically important.	2
SENTENCE_HITS_REQUIRED	The number of hits required for a statement to be considered significant.	1
SENTENCE_DIFFS_ALLOWED	The number of words by which any two sentences may differ and still be considered identical.	0
Other options		
UNKNOWN_WORDS_FILE	The file where the list of words not in the dictionary will be saved. The user may use this list and add the unknown words to the user dictionary.	unknown-words.txt
AUTOMATIC_PATTERNS	Switches on/off the automatic pattern generation. If you switch this off only patterns in the user pattern file will be considered.	ON

Table B.1: Configuration parameters

B.2 Tag set file

The tag set file lists the grammatical tags used in the part-of-speech analysis. It has a simple format: the overall number of tags used in the analysis followed by the list of tags. In this file, each tag must be unique.
Although the tag list can be obtained by studying all the tags present in the dictionary, we require the tag list in advance. This helps detect errors in the lexicon (unknown tags) and also speeds up the algorithm as the lexicon does not need to be read twice (once to determine the tag set used and once to read the word list).

The standard tag set (default) for the English language (with minor modifications) and their interpretation is given in Appendix C.

B.3 Dictionary (lexicon) files

The dictionary (or lexicon) files contain the vocabulary known to the software. There are two such files, the global dictionary (required) and the user dictionary (optional). Both files serve similar purpose and have the same format: a word followed by a list of possible tags on every line, for example,

```
design NN VB
the DT
. .
play VB NN
twice RB JJ
or CC
good JJ NN RB
she PRP
...
```

In the above example, the word 'design' can be a noun (NN) or a verb (VB). The word entries are case sensitive, for example 'design' and 'Design' are considered to be different words. This is because a capitalized word may have a smaller set of possible tags and this information gives a more precise tagging. For instance, the noun 'play' can never appear at the beginning of a sentence, so the dictionary entry for 'Play' does not include the tag NN. When the text is analysed and every word is looked up in the dictionary, the following procedure is performed: If an exact (case sensitive) match is found then the given tags are used; otherwise the software tries to find the word in the dictionary in case-insensitive mode. If both the above operations fail, then the word is added to the list of unknown words (not in the dictionary) and the program continues assuming that the word can take any possible tag.
If the same word appears more than once in the dictionary, the first occurrence of the word is taken into account. If the same word appears in the global and in the user dictionary, the entry in the global dictionary has precedence.

B.4 Substitution file

The purpose of the substitution file is to expand abbreviations to full words. This file is mandatory (although it may be blank), and the default version contains the most common abbreviations of the English language.
The general format requires having one substitution per line, with words separated by spaces, i.e.,

```
<word A> <substitutiotion set {W} of words>
```

During preprocessing, if the first word A is encountered then it is replaced with the sequence of words {W} that follow it. If a transformation requires the addition of a whitespace at the beginning of the replacement string, the special character "~" can be used as a distinct marker to separate the resulting string from preceding words. Instances of "~" will be converted to spaces in the second stage of preprocessing.
Note that substitutions do not have a cumulative effect, that is, if you specify that a word X should be substituted by word Y and later that word Y should be substituted by Z then instances of X will not be changed to Z.
Words specified in this file need not be in the dictionary. However, if after the substitutions have taken place, the text contains unknown words, this will be reported to the user.

B.5 Synonym file

The synonym file (optional) contains a list of words that are considered identical for the propose of the analysis. The use of this file is to reduce the amount of output statements by merging together entries that are identical in meaning but have slightly different wording. The user specifies two or more words that are considered to be synonyms. For example, this file may specify that the words 'truck' and 'lorry' are synonyms, and so are the words 'street' and 'road'. In this case the statements 'the truck is on the road' and 'the lorry is on the street' will be considered identical and their scores in the output will be added together. The user should be careful to specify words that are really synonyms and of the same grammatical category. This is because the software will replace the synonym words in the output with the first word of that synonym group.

Format of the synonym file: a list synonym words per line, each line containing a list of words separated by spaces (i.e., any word in that line is considered to be a synonym with any other word in the same line).

Words specified in the synonym file must exist in the global or in the user dictionary.

B.6 Ignored verbs file

This is an optional file. This simply specifies a list of verbs (one verb per line) that should be ignored in the automatic pattern search procedure. This helps to eliminate sentences in which the main verb (such as 'like' or 'think') is not meaningful unless followed by another verb which most likely contains significant information. In this case the output statement will reflect the context associated with the second verb and the first part of the statement (associated with the first verb) will be ignored.
The usefulness of this file clearly depends on how many of the authors of the messages will use similar 'flavour' or style of writing in their messages.

B.7 Pattern file

This is an optional file specifying the list of grammatical patterns (sequences) that the user wants to include in the analysis. The software has its own procedure of determining which patterns are grammatically significant based on statistical analysis, but this file gives the user the option to specify a list of patterns. The user patterns will be added to the automatically generated ones by the software, unless the option AUTOMATIC_PATTERNS is switched off in the configuration file, in which case only the user-specified patterns will be considered.
Format of the pattern file: one pattern per line, each pattern consisting of a list of 3 to 8 grammatical tags. For example, the pattern

```
DT NN VBZ
```

corresponds to any sequence of 'determinant', 'singular noun' and 'verb', and the analysis will try to find such structures in the corpus file, for example the sentence 'the boy plays' matches the above pattern.
Although not required, it is a good idea to include a verb (VB, VBP, VBZ, or VBD) in the pattern in order to get meaningful sentences.

B.8 System parameter settings

Table B.2 specifies the maximum values set for all system program parameters.

Parameter (length in characters)	***value***
Maximum tag length	6
Maximum word length	40
Maximum sentence length	2000
Maximum length of a line in dictionary file	200
Maximum length of a line in substitution file	100
Maximum length of a line in pattern file	100
Maximum number of unknown words present	1000
Maximum number of user patterns	50
Maximum number of sentences to be output	10000
Maximum characters of text for each message in input database	65000
Maximum characters of text for the messageID column in output database	200
Maximum number of words for each sentence stored in output database	12

Table B.2: System program parameters

APPENDIX C: Grammatical tag set

The standard Penn tag (e.g. [4]) for the English language is presented in the following table.

APPENDIX D: An Illustrative Example

In this Appendix, an example is used to demonstrate how the Intelligent Agent works and its efficiency in automating the process of extracting key information from a collection of user responses. We used an Access database which contained the input, and the same database to store the output. This database is provided along with the Intelligent Agent software for further experimentation.

The example included a survey in which a number of households on the same road were asked to express their views on the following proposal:

"The local council proposes to cut the trees on the pavement of New Road."

The list of responses received is presented in Table D.1.

The 'attitude' column specifies whether the responder agrees with the proposal (0), is indifferent to the proposal (1) or is against the proposal (2). Please note that this information is simply used by the software to classify the output of the analysis into the 'positive', 'negative' and 'neutral' tables. The values of the parameters used in the configuration file for the analysis are:

PATTERN_HITS_REQUIRED	2
SENTENCE_HITS_REQUIRED	1
SENTENCE_DIFFS_ALLOWED	0

Moreover, the default 'global dictionary', 'substitution' and 'ignored verb' files were used and no 'user dictionary', 'user pattern' file or 'synonym' file was specified.

The output produced was stored in two tables in the database, one for the 'positive' output (Table D.2) and one for the 'negative' output (Table D.3). The 'neutral' table was not produced because there were no neutral messages in this example, and also no statement produced appeared as both 'positive' and 'negative'.

The screen output of the program is also shown in figure D.1.

Appendix C: Standard Penn tag set for the English Language e.g. [4]

Category	***Examples***	***Tag***	***Category***	***Examples***	***Tag***
Adjective	happy, bad	JJ	Verb, base present form (not infinitive)	take, live	VBP
Adjective, ordinal number	sixth, 72nd, last	JJ	Verb, infinitive	take, live	VB
Adjective, comparative	happier, worse	JJR	Verb, past tense	took, lived	VBD
Adjective, superlative	happiest, worst	JJS	Verb, present participle	taking, living	VBG
Adjective, superlative, semantically	chief, top	JJ	Verb, past/passive participle	taken, lived	VBN
Adjective, cardinal number	3, fifteen	CD	Verb, present 3SG –*s* form	takes, lives	VBZ
Adjective, cardinal number, one	one	CD	Verb, auxiliary *do*, base	do	VBP
Adverb	often, particularly	RB	Verb, auxiliary *do*, infinitive	do	VB
Adverb, negative	not, n't	RB	Verb, auxiliary *do*, past	did	VBD
Adverb, comparative	faster	RBR	Verb, auxiliary *do*, present part.	Doing	VBG
Adverb, superlative	fastest	RBS	Verb, auxiliary *do*, past part.	Done	VBN
Adverb, particle	up, off, out	RP	Verb, auxiliary *do*, present 3SG	does	VBZ
Adverb, question	when, how, why	WRB	Verb, auxiliary *have*, base	have	VBP
Adverb, degree & question	how, however	WRB	Verb, auxiliary *have*, infinitive	have	VB
Adverb, degree	very, so, too	RB	Verb, auxiliary *have*, past	had	VBD
Adverb, degree postposed	enough, indeed	RB	Verb, auxiliary *have*, present part.	Having	VBG
Adverb, nominal	here, there, now	RB	Verb, auxiliary *have*, past part.	Had	VBN
Conjunction, coordination	and, or	CC	Verb, auxiliary *have*, present 3SG	has	VBZ
Conjunction, subordination	although, when	IN	Verb, auxiliary *be*, infinitive	be	VB
Conjunction, complementizer *that*	that	IN	Verb, auxiliary *be*, past	were	VBD
Determiner	this, each, another	DT	Verb, auxiliary *be*, past, 3SG	was	VBD
Determiner, pronoun	any, some	DT	Verb, auxiliary *be*, present part.	Being	VBG
Determiner, pronoun, plural	these, those	DT	Verb, auxiliary *be*, past part.	Been	VBN
Determiner, prequalifier	quite	PDT	Verb, auxiliary *be*, present 3SG	is, 's	VBZ
Determiner, prequantifier	all, half	PDT	Verb, auxiliary *be*, present 1SG	am, 'm	VBP
Determiner, pronoun or double conj.	Both, either, neither	DT or CC	Verb, auxiliary *be*, present	are, 're	VBP
Determiner, article	the, a, an	DT	Verb, modal	can, could, 'll	MD
Determiner, pastdeterminer	many, some	JJ	Infinitive marker	to	TO
Determiner, possessive	their, your	PRP$	Preposition, to	to	TO
Determiner, possessive, second	mine, yours	PRP	Preposition	for, above	IN
Determiner, question	which, whatever	WDT	Preposition, of	of	IN
Determiner, possessive & question	whose	WP	Possessive	's	POS
Noun	aircraft, data	NN	Interjection (or other isolate)	Oh, yes, mmm	UH
Noun, singular	woman, book	NN	Punctuation, sentence ender	. ! ?	.
Noun, plural	women, books	NNS	Punctuation, semicolon	;	.
Noun, proper, singular	London, Michael	NNP	Punctuation, colon or ellipsis	: …	.
Noun, proper, plural	Australians, Methodists	NNPS	Punctuation, comma	,	,
Noun, adverbial	tomorrow, home	NN	Punctuation, dash	-	SYM
Noun, adverbial, plural	Sundays, weekdays	NNS	Punctuation, dollar sign	$	SYM
Pronoun, nominal (indefinite)	none, everything, one	NN	Punctuation, left bracket	([{	(
Pronoun, personal, subject	you, we	PRP	Punctuation, right bracket	)] }	)
Pronoun, personal, subject, 3SG	she, he, it	PRP	Punctuation, quotation mark, left	‘ “	“
Pronoun, personal, object	you, them, me	PRP	Punctuation, quotation mark, right	’ ”	”
Pronoun, reflexive	herself, myself	PRP	Foreign words (not in English lexicon)		FW
Pronoun, reflexive, plural	themselves, ourselves	PRP	Symbol	*	SYM
Pronoun, question, subject	who, whoever	WP	Symbol, alphabetical	A, B, c, d	SYM
Pronoun, question, object	who, whoever	WP	Symbol, list item	A A. First	
Pronoun, existential, *there*	there	EX			

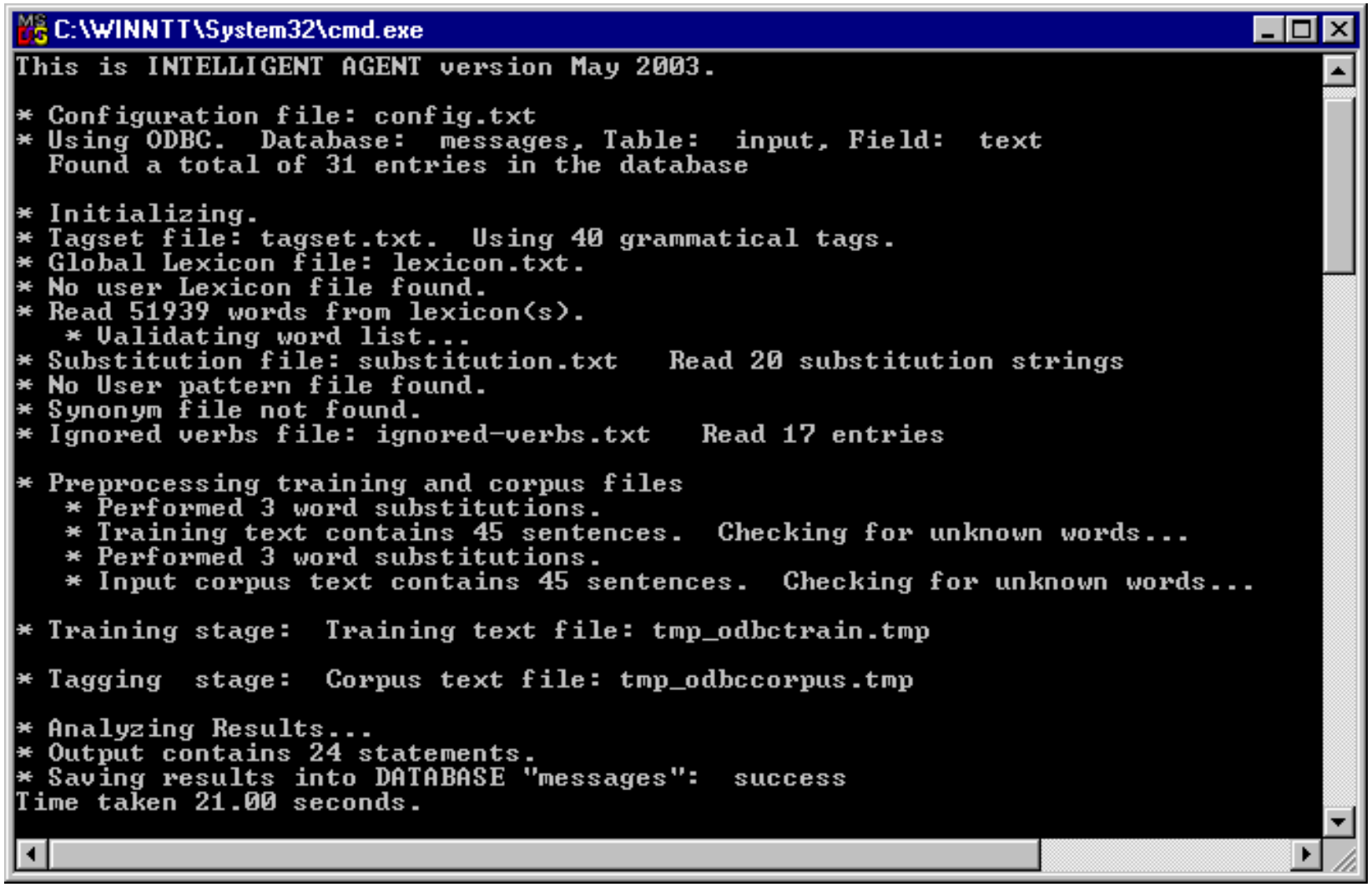


Figure D.1: Screen output of the Intelligent Agent execution

ID	*attitude*	*message text*
1	2	Trees are necessary. They make the road look beautiful. I would vote against cutting the trees.
2	0	Cutting the trees? Yes, certainly. I do not think they are necessary.
3	0	Trees cause problems when parking my car. My car gets dirty sometimes because birds are sitting on the trees.
4	0	The road becomes full of leaves, so eliminate the trees.
5	0	Please cut the trees. They can cause damage to the cars when it is windy in the winter.
6	0	Replace the trees with evergreen.
7	0	Trees can cause subsidence of nearby houses, so better cut them.
8	0	Trees can have deep roots which can break up the foundations. I think you should reduce the number of trees on the road.
9	2	Trees are a source of oxygen and, therefore, they are absolutely necessary for us.
10	2	The trees are a natural habitat for birds. I would not like to be cut.
11	2	I would never propose against cutting the trees because they are a natural wind barrier.
12	2	I have been living here for 50 years and the trees have always been there. I strongly oppose this proposal.
13	2	I do not want to see them go because I like sitting in the shadow when it gets hot in the summer.
14	0	I want the trees to go because their flowers and leaves fall on my garden.
15	0	What a headache! I need to remove the leaves from my front garden every day in autumn. I would be happy to get rid of this task.
16	2	I cannot imagine the neighbourhood without this natural beauty and tranquillity.
17	2	Why cut the trees? They are a natural source of oxygen.
18	2	Leave the trees as they provide a source of oxygen in our neighbourhood.
19	0	Cut the trees because their roots cause damage to the houses.
20	2	The trees protect us from the noise of the street, so leave them.
21	0	The roots of the trees cause damage to our houses.
22	0	Remove the trees as their leaves fall on my car.
23	2	The trees have been here for more than 40 years so I cannot see a reason why they should be removed.
24	2	The trees are a wind barrier and I would like them to remain here.
25	0	The leaves fall on my garden and I would be very happy to see them go.
26	0	I do not think the trees are necessary, please cut them.
27	2	I like to sit in their shadow in the summer.
28	2	I enjoy the tranquillity they provide in the summer.
29	2	The trees are a source of oxygen and are good for the neighbourhood.
30	0	I have to remove the leaves every day. I will be happy to see them go.
31	2	The trees offer cover from the noise of the street, this benefit will be removed if they are cut. I don't want to see them go.

Table D.1: List of messages received from the survey

frequency	messageID	pattern	votes	length	word1	word2	word3	word4	word5	word6	word7	word8	word9	word10	word11	word12
2	25 30	43	2	5	happy	to	see	them	go							
2	15 30	29	2	4	to	remove	the	leaves	from	my	front	garden	every	day	in	autumn
1	30	48	1	4	I	will	be	happy	to	see	them	go				
1	22	37	1	5	Remove	the	trees	as	their	leaves	fall	on	my	car		
1	19	35	1	5	Cut	the	trees	because	their	roots	cause	damage	to	the	houses	
1	15	30	1	4	I	would	be	happy	to	get	rid	of	this	task		
1	8	7	1	3	can	have	deep	roots	which	can	break	up	the	foundations		
1	5	3	1	7	when	it	is	windy	in	the	winter					
1	5	2	1	4	damage	to	the	cars	when	it	is	windy	in	the	winter	
1	4	1	1	3	road	becomes	full	of	leaves	,	so	eliminate	the	trees		
1	3	0	1	3	car	gets	dirty	sometimes	because	birds	are	sitting	on	the	trees	

Table D.2: The 'positive' output

frequency	messageID	pattern	votes	length	word1	word2	word3	word4	word5	word6	word7	word8	word9	word10	word11	word12
2	13 31	12	2	8	I	do	not	want	to	see	them	go				
2	9 29	9	2	5	are	a	source	of	oxygen							
1	29	46	1	5	The	trees	are	a	source	of	oxygen					
1	28	45	1	4	I	enjoy	the	tranquillity	they	provide	in	the	summer			
1	24	41	1	5	The	trees	are	a	wind	barrier						
1	23	38	1	6	I	can	not	see	a	reason						
1	18	34	1	5	provide	a	source	of	oxygen	in	our	neighbourhood				
1	18	33	1	4	they	provide	a	source	of	oxygen	in	our	neighbourhood			
1	17	32	1	4	are	a	natural	source	of	oxygen						
1	13	24	1	7	when	it	gets	hot	in	the	summer					
1	11	11	1	4	are	a	natural	wind	barrier							
1	10	10	1	4	are	a	natural	habitat	for	birds						
1	9	8	1	4	Trees	are	a	source	of	oxygen						

Table D.3: The 'negative' output.

REFERENCES

[1] E. Brill and M. Marcus. *Tagging an unfamiliar text with minimal human supervision*. Department of Computer and Information Science, University of Pennsylvania, 1992.

[2] R. Grishman. *Computational Linguistics: an Introduction*. Cambridge, 1986.

[3] B. Krenn and C. Samuelson. *The linguist's guide to statistics*. Bell Labs, 1997.

[4] C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT press, 2000.

[5] L. R. Rabiner. *A tutorial on Hidden Markov Chains and selected applications in speech recognition*. Proceedings of the IEEE, vol 77 (2), 1989.

[6] O. Rambow. *Introduction to syntax with part-of-speech tagging*. Department of Computer Science, Columbia University, 2002.

ANNEX 4

Screenshots of the user interface and it´s modules

Annex 4 contains in itself several screenshots of the user interface, that give a better picture from the future eCommunity System.

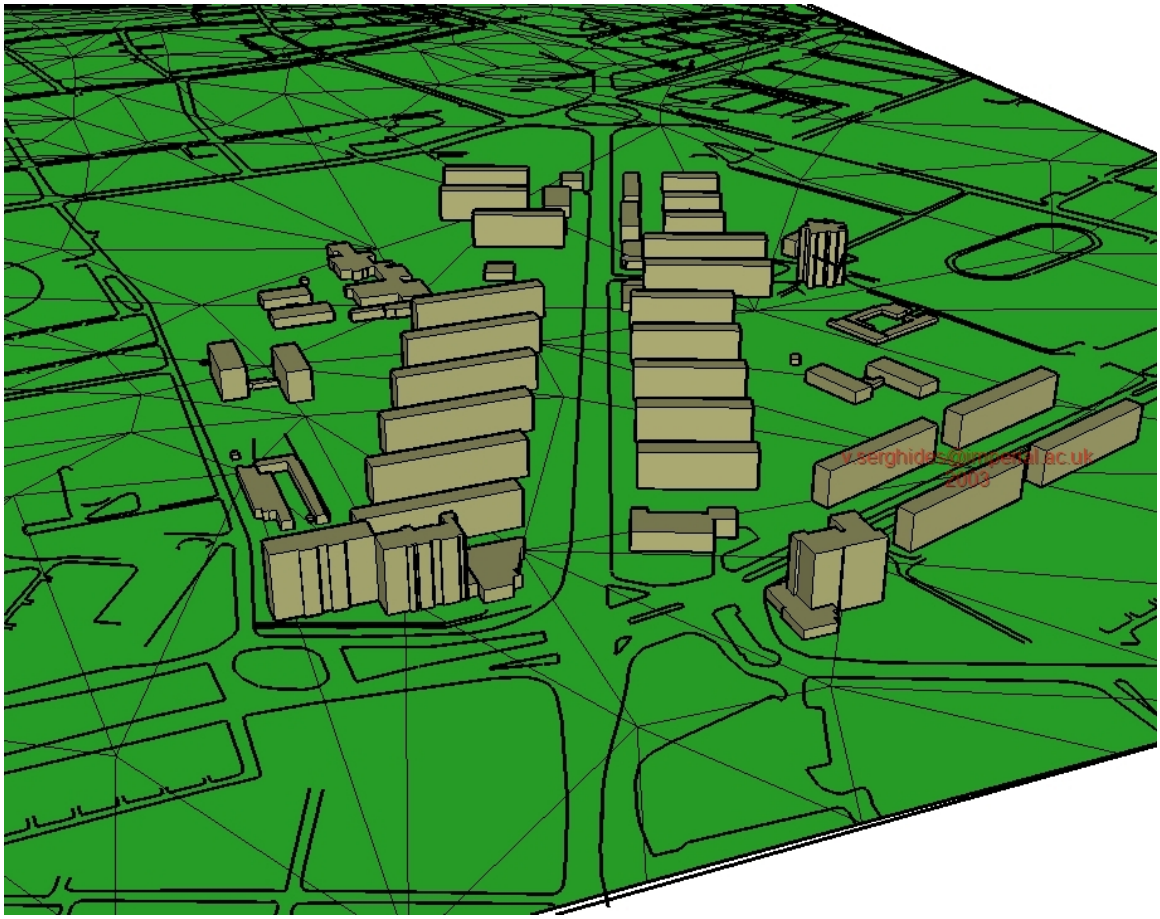


Figure 1. Screenshot no.1 of the future 3D module

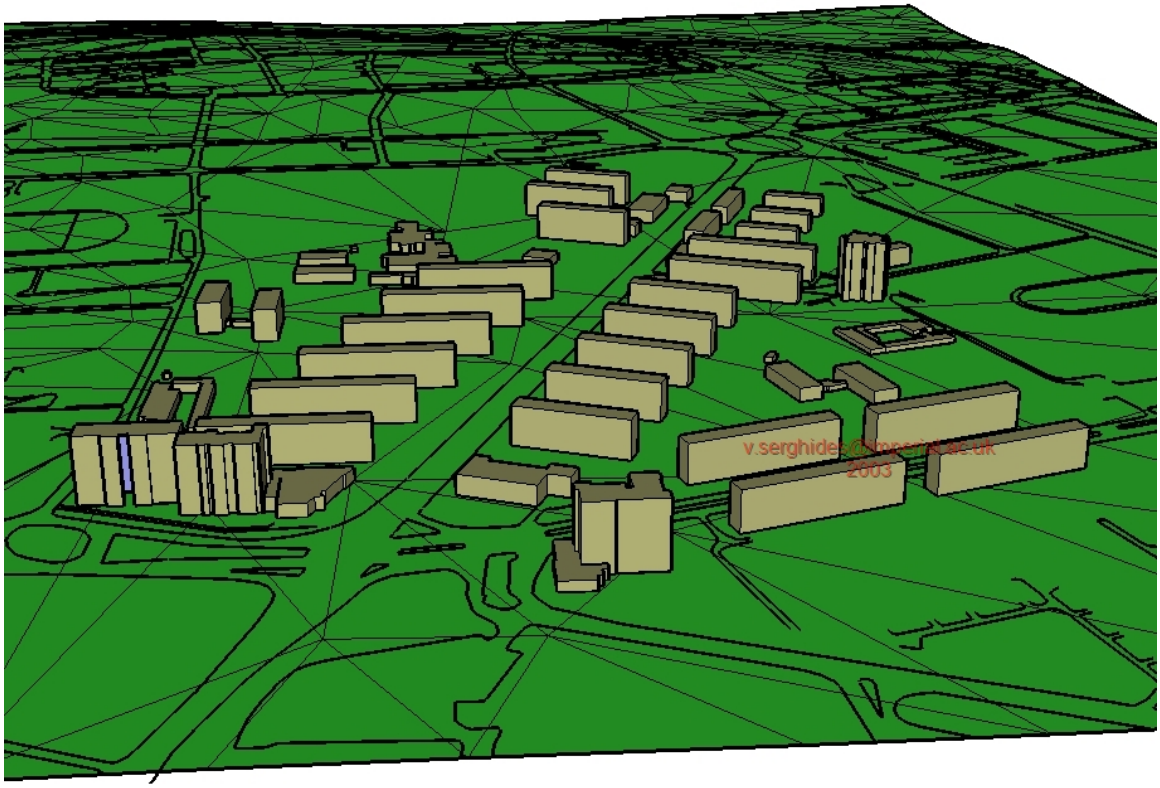


Figure 2. Screenshot no.2 of the 3D module

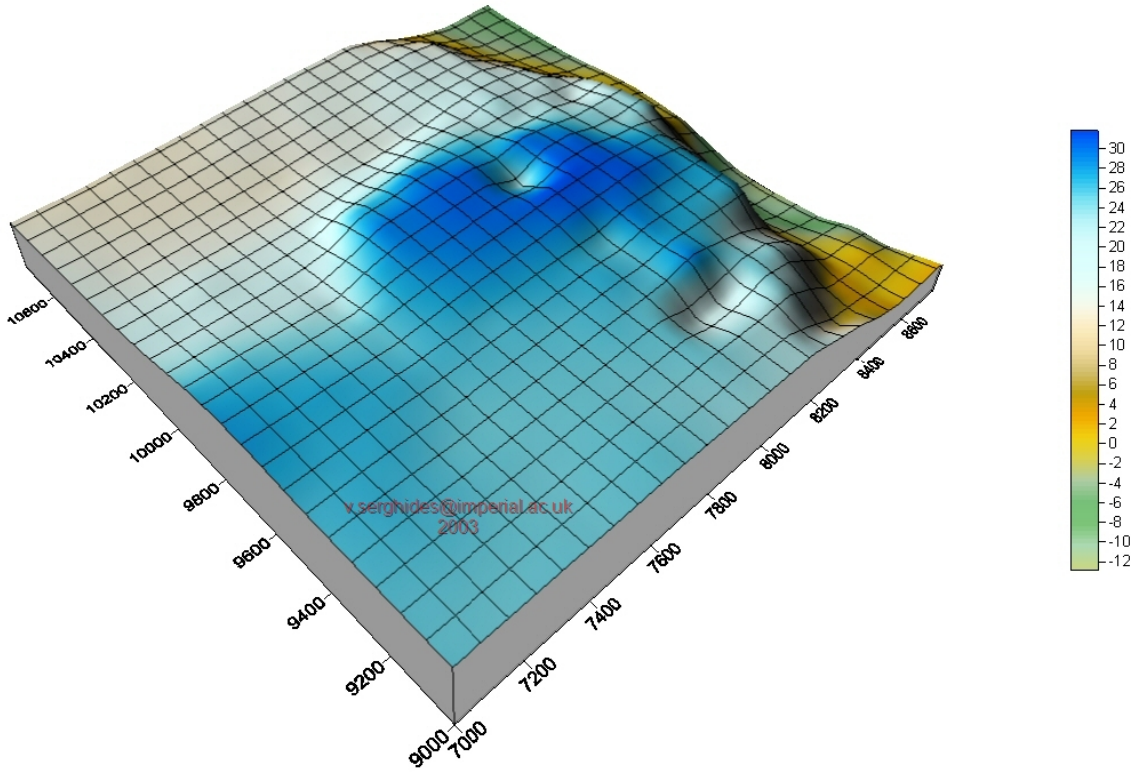


Figure 3. Terrain of the 3D Narva City module

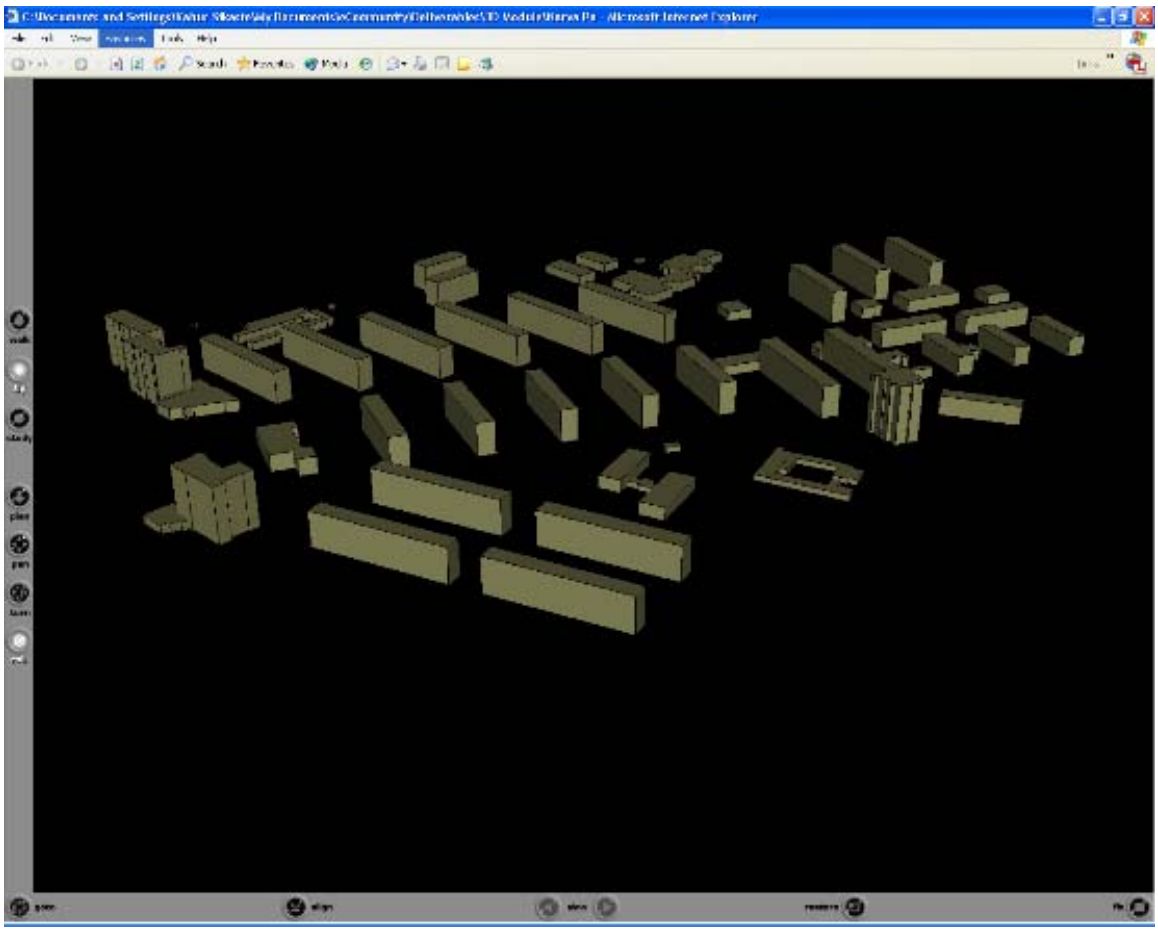


Figure 4. Screenshot no. 3 of the future 3D module

Secure user registration.

Your id-card gives you a unique id within the system, but we still need some personal data that cannot (currently) be automatically retrieved. Please fill in the following form carefully with the necessary data. You only need to do this once. All fields are mandatory.

Make sure you choose the right language, because it will be used for displaying all subsequent pages (polls etc.), and also for interpreting your votes.

Important:

All Estonian citizens have an id-card but only Narva residents are allowed to use it for accessing this site. The information you enter here will be verified by the Municipality of Narva. If you are not a Narva resident, you have to visit this site again without your id-card and choose a username/password for accessing it normally.

Name (first and last name)	
Address	
E-mail	
Language	Estonian
>>Continue	

Figure 5. Screenshot of the user management module

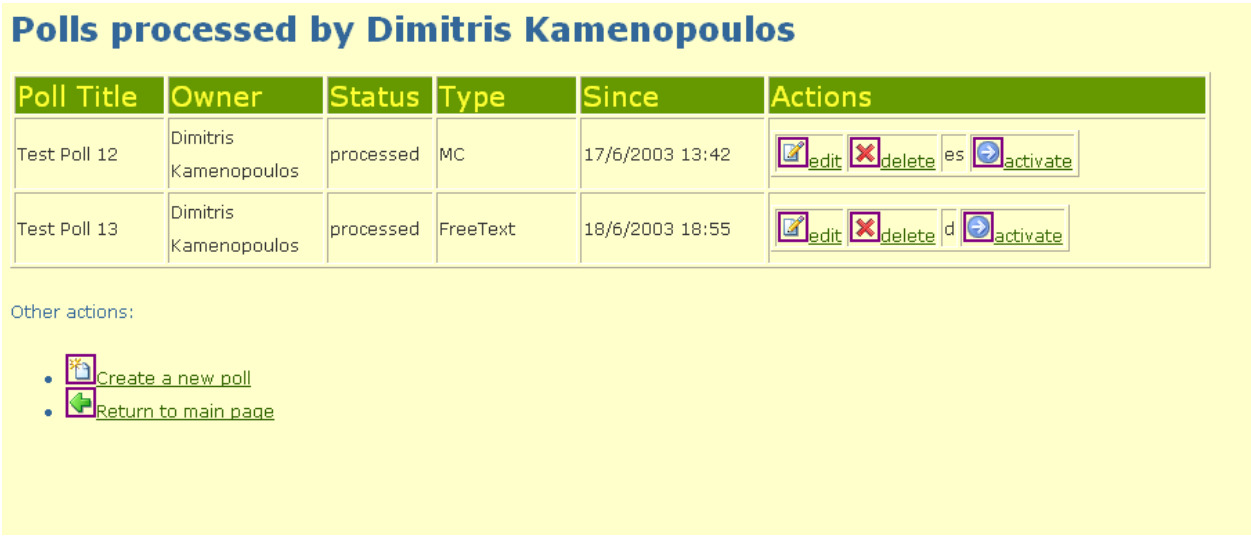


Figure 6. Screenshot of the DMM managament module

ANNEX 5

X-TEE requirements

X-tee

Protocol: Data exchange protocol between dataset and information system

Requirements on information systems and adapter servers

Version 6.1

Contents

1. Introduction 5

2. Concepts and notions 6
- 2.1 Standards used 6
  - 2.1.1 WSDL 6
  - 2.1.2 SOAP 6
  - 2.1.3 XMLRPC 6
- 2.2 Namespaces 6
- 2.3 Meaning of queries 6

3. Structure of messages in data exchange protocol 7
- 3.1 Division of data between components 7
- 3.2 Elements of the header component 7
- 3.3 Message presentation in SOAP 8
  - 3.3.1 Parameter presentation 8
  - 3.3.2 Query input 8
  - 3.3.3 Query output 9
- 3.4 Message presentation in XMLRPC 9
  - 3.4.1 Parameter presentation 9
  - 3.4.2 Query input without header and body 9
  - 3.4.3 Query input with header and body 10
  - 3.4.4 Query output without header, query and body 10
  - 3.4.5 Query output with header, query and body 10
- 3.5 Error messages 11
  - 3.5.1 SOAP standard error message 11
  - 3.5.2 XMLRPC standard error message 12
  - 3.5.3 SOAP non-technical error message 12

4. Description format of data queries 12
- 4.1 Names of queries and parameters 12
- 4.2 WSDL format 13
- 4.3 Publication of query descriptions 14

5. Metaqueries 16
- 5.1 Overview 16
- 5.2 Description of metaqueries 16
  - 5.2.1 Query for list of datasets listProducers 16
  - 5.2.2 Query for methods allowed for an institution allowedMethods 16
  - 5.2.3 Query for adapter server's methods listMethods 17
  - 5.2.4 Query for money charged for a query getCharge 17
  - 5.2.5 Query for loading a classifier loadClassificator 18
  - 5.2.6 Query for entering a legacy system legacyXYZ 18

6. References 20

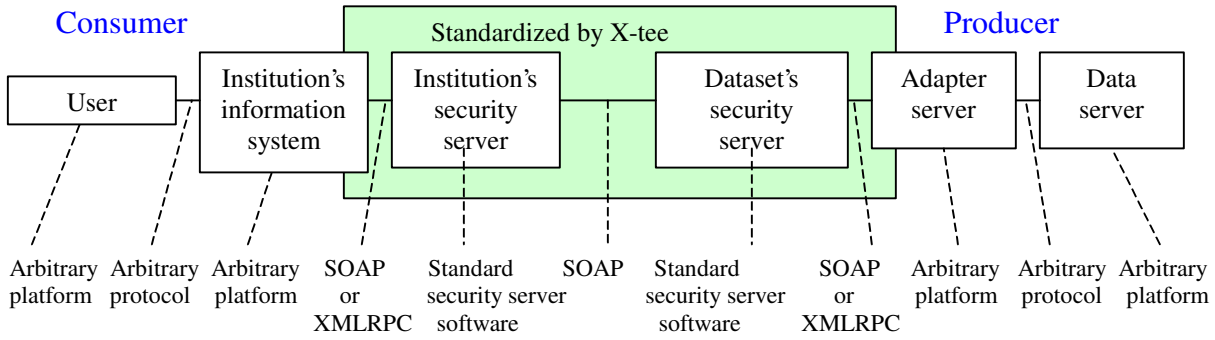
7. Examples 21
- 7.1 SOAP examples 21
  - 7.1.1 Query listProducers 21

7.1.2 Query allowedMethods 22
7.1.3 Query listMethods 23
7.1.4 Data query 24
7.1.5 Query andmekogu.loadClassificator 26
7.2 XMLRPC examples 29
7.2.1 Query listProducers 29
7.2.2 Query allowedMethods 31
7.2.3 Query listMethods 34
7.2.4 Data query 35
7.3 WSDL example 40

Requirements on information systems and adapter servers

1. Introduction

The data exchange protocol of X-tee (*X-Road*) regulates data exchange between an information system and the adapter server of a dataset. The information system of an institution provides service to end users, utilizing platforms and protocols independent of X-tee. The X-tee-related communication of both the information system and adapter server occurs only via corresponding security servers. The security server of an institution makes connection to the security server of the dataset and forwards the query received from the information system. The adapter server of the dataset modifies queries arriving from X-tee to a form that can be processed by the dataset's data server (which is independent of X-tee), and returns the data server's response in a form suitable for X-tee.



Either SOAP or XMLRPC protocol can be used in the communication between institution's informaton system and security server, as well as between dataset's adapter server and security server. The infosystem's ability to access the dataset via X-tee doesn't depend on the protocols used by either security server. If necessary, conversion between SOAP and XMLRPC will be performed by security server.

Adapter servers and information systems can be implemented by any software vendor. The only technical requirement presented by X-tee is this: the software has to communicate with security server via standardized X-tee protocol. Institutions's information system must also have sufficient security level to be connected to X-tee.

Implementing an adapter server means creating a SOAP or XMLRPC server and publishing necessary data queries, as well as metaqueries obligatory for adapter servers. The parameters of adapter server are stored in the dataset's security server which will access the adapter server.

For integrating an institution's information system with X-tee, a SOAP or XMLRPC client has to be created for forwarding queries. The information system with SOAP client will pass all queries via http or https to an URL in the institution's security server: /cgi-bin/consumer_proxy . The information system with XMLRPC client will pass all queries via http or https to an URL in the institution's security server: /cgi-bin/consumer_proxy_xmlrpc .

Standard libraries are usually used for creating SOAP or XMLRPC servers and clients. A set of SOAP libraries can be found at http://www.soapware.org/directory/4/implementations . A set of XMLRPC libraries can be found at http://www.xmlrpc.com/directory/1568/implementations .

This document describes the restrictions that X-tee places on standard protocols, the types of queries that can be used/implemented on X-tee, and how these queries should be described..

2. Concepts and notions

2.1 Standards used

2.1.1 WSDL

WSDL [WSDL] is a language for describing Web services. X-tee queries conform to the WSDL specification, with restrictions that will be described in this document.

2.1.2 SOAP

SOAP [SOAP] is an XML-based data exchange protocol. Information systems and adapter servers can communicate with security servers via protocol that conforms to the SOAP specification, with restrictions that will be described in this document.

2.1.3 XMLRPC

XMLRPC [XMLRPC] is an XML-based remote service protocol. Information systems and adapter servers can communicate with security servers via protocol that conforms to the XMLRPC specification, with restrictions that will be described in this document.

2.2 Namespaces

This document uses the following namespace prefixes for referring to namespaces.
Prefix *xtee* refers to X-tee namespace http://x-tee.riik.ee/xsd/xtee.xsd.
Prefix SOAP-ENV refers to namespace http://schemas.xmlsoap.org/soap/envelope/.
Prefix SOAP-ENC refers to namespace http://schemas.xmlsoap.org/soap/encoding/.

2.3 Meaning of queries

The SOAP or XMLRPC methods executed on X-tee form X-tee queries. An X-tee query is a two-stage data exchange between information system and dataset, initiated by the information system by sending a query (a message with query input), which is then processed by the dataset to return query results (a message with query output). From technical viewpoint, it is irrelevant for X-tee whether the aim of the query is to generate a report of dataset's data based on the conditions given in query input, to store the data from query input into the dataset, or something else.

Queries can be divided into data queries and metaqueries.
Data queries are queries specific to a particular dataset and usually created individually for that dataset. To give access to these queries is the main goal of X-tee. The input and output of data queries of a dataset are specified in the documentation of the dataset.
Metaqueries are auxiliary queries for obtaining information necessary to perform data queries. The input, output and semantics of metaqueries are standardized and will be described in this document. They are similar in all servers providing metaqueries.
Metaqueries belong to X-tee namespace http://x-tee.riik.ee/xsd/xtee.xsd.
Data queries belong to the namespace of their corresponding dataset:
http://producers.**dataset**.xtee.riik.ee/producer/**dataset** , where **dataset** is the name of the dataset.

3. Structure of messages in data exchange protocol

3.1 Division of data between components

Data passed over X-tee is divided between the following components: header, query and body. Their exact representation depends on protocol. A component usually consists of sub-parameters, but it can also contain a scalar value.

A message with query input contains the following components: header and body.

A message with query output contains the following components: header, query and body. Here, the contents of the header component are identical to the header component of the query input message, and the contents of the query component are identical to the body component of the query input message. In other words, the header component is copied from input message to output message, and the input message's body is copied to the query component of the output message.

In case of SOAP, the header component is presented inside the SOAP envelope header. The query and body components are presented as elements <paring> and <keha> immediately below the root element of SOAP envelope body.

In case of XMLRPC, the components of query input are presented as anonymous parameters in the order *header*, *query*. The components of query output are presented as named parameters, with the following names: *pais* for the header component, *paring* for the query component, *keha* for the body component.

All messages passed over X-tee are in the UTF-8 encoding.

3.2 Elements of the header component

Query header is a structure with the following elements:

```
Struct                      -- header
        String asutus       -- name of the institution
        String andmekogu    -- name/prefix of the dataset
        String ametnik      -- ID code of the person executing the query
        String id           -- query nonce
        String nimi         -- name of the method to be called
        String toimik       -- name of file (or document) related to the query
```

Query nonce is a unique identifier consisting of numbers and letters of Latin alphabet. The identifier is generated by the information system that initiated the query. It is the responsibility of the information system to guarantee that the identifier is globally unique. For example, the identifier could contain a sufficiently large random number, or a random number, name of the institution and query checksum. The information system could use the query identifier for associating the query with the data it was based on (eg a person's application).
The method name in the header must match the name of the method that is executed. It is duplicated in order to facilitate processing of signed queries.
In addition to the elements listed above, the header of a query can also contain other header elements described in the X-tee namespace:

```
        String makstud      -- the sum of money (in cents) paid for executing the query
        String amet         -- the post of the person initiating the query
```

String allasutus -- register code of the institution on whose behalf the query is executed (used in the portal for enterprises only)

All the header elements mentioned above belong to the X-tee namespace.

3.3 Message presentation in SOAP

In this section, the elements that contain values specific to the query are printed in **boldface**.
Texts describing context for which there is no syntax provided are printed in ***italic***.
The attribute encodingStyle can only be used in the element <SOAP-ENV:Envelope> and its value must always be http://schemas.xmlsoap.org/soap/encoding/.

3.3.1 Parameter presentation

In parameter presentation, **name** is name of the parameter, **type** is type of the parameter and **value** is value of the parameter. The name and type of the parameter are given in the description of the parameter; the value of the parameter is either a scalar (in case of scalar parameters) or representations of sub-parameters (in case of structured parameters). In the header and body components, namespace prefixes are not used, because all elements are defined by the namespace of the sub-parameter (**<method>** või **<method**Response>) of the SOAP envelope body (<SOAP-ENV:Body>).
NB! Specifying parameter type is obligatory.

<name type=”**type**”>
value
</name>

In an X-tee message, all elements must be presented as single-reference. Polymorphic elements cannot be used, ie all parameters in a SOAP message must be accessible via only one path that defines the parameter uniquely.

In the following, the texts "contents of header component", "contents of query component", "contents of body component" mean a SOAP representation of all sub-parameters of corresponding component (if the component is a structured parameter) or the component's value (if the component is a scalar parameter). Elements can have other attributes besides those listed here.

3.3.2 Query input

A message containing query input has the following structure.
Element <keha> together with the contents of the body component may be missing, if the query to be executed is a metaquery without parameters.

```
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Header>
        contents of header component
     </SOAP-ENV:Header>
    <SOAP-ENV:Body>
        <m:method xmlns:m=”URI”>
            <keha>
                contents of body component
            </keha>
        </m:method>
    </SOAP-ENV:Body>
```

```
</SOAP-ENV:Envelope>
```

3.3.3 Query output

A message containing query output has the following structure.
Element <paring> together with the contents may be missing, if the query input didn't contain the element <keha>.

```
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" >
  <SOAP-ENV:Header>
      contents of the header component
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
      <m:methodResponse xmlns:m="URI">
          <paring>
            contents of the query component
          </paring>
          <keha>
            contents of the body component
            </keha>
      </m:methodResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3.4 Message presentation in XMLRPC

In this section, the elements that contain values specific to the query are printed in **boldface**.
Texts describing context for which there is no syntax provided are printed in ***italic***.

3.4.1 Parameter presentation

In parameter presentation, **name** is name of the parameter, **type** is type of the parameter and **value** is value of the parameter. The name of the parameter is given in the description of the parameter; type is the equivalent among XMLRPC types of the type given in query description; the value of the parameter is either a scalar (in case of scalar parameters) or representations of sub-parameters (in case of structured parameters). Element <name> together with its contents exists in case of named parameters (if the name is present in the query description) and is missing in case of anonymous parameters (if there is no name in the query description). Element **<type>** can be missing in cases allowed by the XMLRPC standard.

```
<member>
  <name>name</name>
  <value><type>value</type></value>
</member>
```

In the following, the contents of the header, query and body components are either anonymous (only value for query input) or named (name and value for query output) XMLRPC representations of corresponding components.

3.4.2 Query input without header and body

Header and body are missing only in the methods *(xtee.)listProducers* ja *(system.)listMethods*.

```
<methodCall>
   <methodName>
      dataset.method
   </methodName>
   <params>
      <param>
         <value>
            <string/>
         </value>
      </param>
   </params>
</methodCall>
```

3.4.3 Query input with header and body

Message with query input usually has the following structure.

```
<methodCall>
   <methodName>
      dataset.method.version
   </methodName>
   <params>
      <param>
         contents of the header component
      </param>
      <param>
         contents of the query component
      </param>
   </params>
</methodCall>
```

3.4.4 Query output without header, query and body

Header, query and body are missing only in the methods *(xtee.)listProducers* ja *(system.)listMethods*.

```
<methodResponse>
   <params>
      <param>
         <value>
            parameters of response
         </value>
      </param>
   </params>
</methodResponse>
```

3.4.5 Query output with header, query and body

Message with query output usually has the following structure.

```
<methodResponse>
   <params>
      <param>
```

```
            <value>
                <struct>
                    <member>
                        contents of the header component
                    </member>
                    <member>
                        contents of the query component
                    </member>
                    <member>
                        contents of the body component
                    </member>
                </struct>
            </value>
        </param>
    </params>
</methodResponse>
```

3.5 Error messages

The output of a query can be an error message. Error messages can be presented in two ways, depending on the contents of the message.
If the error message reports that a technical problem not related to the user does not allow to perform the query successfully, then the message is presented in standard SOAP or XMLRPC error message format.
If the error message concerns input data submitted by the user (eg a data query returns no results), then the message is presented as a <keha> (body) structure with the following elements: *faultCode* (error code) and *faultString* (error message).

3.5.1 SOAP standard error message

If possible, a SOAP error message should also contain a header, but it is not obligatory.
Error codes correspond to codes in SOAP specification. Codes belonging to class *Client* refer to errors in the query input composed by the information system. Codes belonging to class *Server* refer to errors not related to query input. Error codes generated by security servers are listed in the document [XVEAD].
The parameter *faultactor* is used to store the identity of the component that created the error message, if possible.

```
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Header>
        contents of the header component
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
      <SOAP-ENV:Fault>
        <faultcode>faultcode</faultcode>
        <faultactor>faultactor</faultactor>
        <faultstring>faultstring</faultstring>
        <faultdetail>faultdetail</faultdetail>
      </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3.5.2 XMLRPC standard error message

```
<methodResponse>
 <fault>
  <param>
   <value>
    <struct>
     <member>
      <name>faultString</name>
      <value>
        <string>
          faultstring
        </string>
      </value>
     </member>
     <member>
      <name>faultCode</name>
      <value>
          <int>faultcode</int>
       </value>
     </member>
    </struct>
   </value>
  </param>
 </fault>
</methodResponse>
```

3.5.3 SOAP non-technical error message

```
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Header>
   pcontents of the header component
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <methodResponse>
      <keha>
        <faultCode xsi:type="xsd:int">faultcode</faultCode>
        <faultString xsi:type="xsd:string">faultstring</faultString>
      </keha>
    </methodResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

4. Description format of data queries

4.1 Names of queries and parameters

A query is identified by its name.
The name of a query consists of three components separated by dots, in the form *dataset.method.version*, where

- *dataset* is the name of the dataset. The name is fixed upon its joining X-tee;
- *method* is short name of the query, unique among this dataset's queries;

- *version* is version of the query in the form v*N*, where *N* is version number.

All three components must conform to the character set allowed in DNS (letters, numbers, underscore, but no national characters).
The short name of a data query cannot match the name of any metaquery.

Example of a query name: buildingsregistry.query2.v1 .

Any individual version of a query is fixed and must not change.
Any change in the description of a query requires creation of either a new version of the query, or, if the semantics of the query changed, creation of a new query. The semantics of a query must be the same across all versions of the query.

Names of query parameters can consist of ASCII letters (ie no national characters), numbers, underscores and dots. The name of a parameter cannot begin with an underscore or a number.

4.2 WSDL format

The WSDL format of query description conforms to the WSDL specification, with the restrictions listed below.

Input and output parameters of queries are described as an XML schema [XSD].
The following elements have been added to WSDL to store information necessary for X-tee.

Element's location (XPath)	**Value**
/definitions/service/port/xtee:address	No value, but attribute *producer*, whose value is the name of the dataset
/definitions/service/port/xtee:title	Heading of dataset (for displaying to the user)
/definitions/binding/operation/xtee:version	Version of query
/definitions/binding/operation/xtee:charge	Payment data (if the query must be paid for in the citizens' portal)
/definitions/binding/operation/xtee:charge/xtee:account	Bank account of the payee
/definitions/binding/operation/xtee:charge/xtee:receivername	Name of payee
/definitions/binding/operation/xtee:charge/xtee:message	Description of payment
/definitions/binding/operation/xtee:charge/xtee:amount	Sum of payment in cents. The element can have attribute *chargeType*, whose value is the type of payee to whom the sum applies
/definitions/portType/operation/documentation/xtee:title	Query heading (for displaying to the user)
/definitions/portType/operation/documentation/xtee:notes	Query comment (for displaying to the user)
/definitions/portType/operation/documentation/xtee:technotes	Query comment (for implementor)
//annotation/appinfo/xtee:title	Parameter heading (for displaying to the user)
//annotation/appinfo/xtee:notes	Parameter comment (for displaying to the user)
//annotation/appinfo/xtee:technotes	Parameter comment (for implementor)
//annotation/appinfo/xtee:fieldtype	Recommended type of input parameter. Only one value is allowed: *textarea*, denoting the <*textarea*></*textarea*> element of HTML
//annotation/appinfo/xtee:fieldrows	Recommended number of rows on screen for the input parameter (the value of attribute *rows* in the HTML element <*textarea rows='...' cols='...'*></*textarea*>)
//annotation/appinfo/xtee:fieldcols	Recommended length of input parameter on screen (the valueof attribute *cols* in the HTML element <*textarea rows='...' cols='...'*></*textarea*>)

//annotation/appinfo/xtee:fieldsize	Recommended length of input parameter on screen (the valueof attribute *size* in the HTML element <*input type='text' size='..'*> atribuudi *size* väärtus)
//annotation/appinfo/xtee:wildcard	A list of metasymbols allowed in parameter value
//annotation/appinfo/xtee:ref	Name of parameter to which this parameter is related

If the element <xtee:address> is present, then the query is considered accessible via X-tee protocol.
In describing X-tee queries, the following elements are obligatory: name and heading of dataset; name, heading and version of query. It is recommended that each parameter is given a name.
In the elements <xtee:title>, <xtee:notes>, <xtee:technotes> the attribute *xml:lang* can be used to store the language in which the value of the element is presented. If the attribute is missing, then the default value 'ee' is presumed.
When issuing queries, metasymbols can be used in parameter values if the metasymbols are listed in the description of the query. In query description, all metasymbols allowed in parameters should be listed in the element <xtee:wildcard> . The following symbols can be used.

*	(asterisk)	Metasymbol * can be used to replace any number of arbitrary symbols
?	(question mark)	Question mark ? can be used to replace one arbitrary symbol
-	(hyphen)	Hyphen can be used to denote interval
P	(letter P)	Value is considered a prefix
S	(letter S)	Value is considered a substring

Regarding parameter types, it should be noted that if the information system initiating the query and the adapter server do not both use the SOAP protocol, then the security server performs a SOAP-XMLRPC translation which can modify some types, because the number of types supported by XMLRPC is considerably less than the number of types supported by SOAP. If derived types are used in query description, then parameter of the derived scalar type may be converted into parameter of the string type no matter what was the base type of the derivation. In case of partially presented array types, the information that data is only partially given may be lost. Therefore, information systems and adapter servers should not handle parameter types very strictly.

To avoid the possibility that an error causes absence of query output and another error causes absence of corresponding error message, it is recommended that the query output is designed in a way that the query body contains at least one non-empty scalar parameter, independent of how the query was called. This parameter could be a non-technical error message.

4.3 Publication of query descriptions

Dataset administrator publishes descriptions of the dataset's data queries as a WSDL-format file in an adapter server accessible to the security server. All data queries of a dataset must be included in one WSDL file with the name *dataset.wsdl* (where *dataset* is name of the dataset). The URL of the file is stored in the settings of the dataset's security server. If necessary, the WSDL file can refer to other files either on public internet or in the same folder as the main WSDL file, thus accessible to the security server.

If the query description file is on a web server accessible to the institution's information system, then the query descriptions can be downloaded with ordinary http GET queries. If the web server is not accessible, then query descriptions can be downloaded via the institutions's security server, as described below.

If the information system acesses the security server at the following URL
http://**securityserver**/cgi-bin/uriproxy?uri=**URI**

where **securityserver** is the address of the security server and **URI** is an absolute URI valid in public internet, then it can download the WSDL or schema file from that location.

If the information system acesses the security server at the following URL
http://**securityserver**/cgi-bin/uriproxy?producer=**dataset**&filename=**filename**
where **securityserver** is the address of the security server, **dataset** is dataset's name and **filename** is a short name (without path) of a WSDL or schema file of that dataset, then it can download file named **filename** from the folder of WSDL files of that dataset.

5. Metaqueries

5.1 Overview

An institutions's information system can access the following metaqueries:

listProducers
allowedMethods

An institutions's information system may also have access to some of the following metaqueries on some datasets:

getCharge
loadClassificator
legacyXYZ[1]

An adapter server must implement the following metaqueries:

listMethods

An adapter server may also implement the following metaqueries:

getCharge
loadClassificator
legacyXYZ

5.2 Description of metaqueries

5.2.1 Query for list of datasets listProducers

Query *xtee.listProducers* returns a list of existing datasets (aka producers). This query allows the information system to automatically find out the names of available datasets, so that later it can query the datasets for the methods (queries) that they provide.

Input body missing

Output body

Array
    Struct
        String name -- dataset's prefix or short name
        String description -- dataset's full name

Output parameter is a list of these structures. Each structure corresponds to one dataset, giving the dataset's name or prefix in member **name**, and an official full name of the dataset in member **description**.

In XMLRPC, this query is called without header, query and body components, ie no parameters in input; output contains one anonymous parameter which is a list of the strcutures described above.

5.2.2 Query for methods allowed for an institution allowedMethods

Query *dataset.allowedMethods* (where *dataset* is dataset's name) is implemented in the dataset's security server and is called by an institution's imformation system. The query returns list of methods acessible to the institution.

Input body is an empty string

String -- query (empty)

[1] XYZ is arbitrary text characteristic to that query

Output body

Array keha -- body of response

String -- name of query in the form *dataset.method.version*

The body component of the method's input is empty. The body component of the method's output is a list of allowed methods.

5.2.3 Query for adapter server's methods listMethods

Method *system.listMethods* implemented in an adapter server returns a list of all methods implemented in the adapter server. This method is meant to be called only by the dataset's own security server.

Input body missing

Output body

Array

String -- name of query in the form *dataset.method.version*

Outpupt parameter is a list of method names.

In XMLRPC, this query is called without header, query and body components, ie no parameters in input; output contains one anonymous parameter which is the list of method names described above.

5.2.4 Query for money charged for a query getCharge

Query *dataset.getCharge* (where *dataset* is dataset's name) is necessary if the service provider (producer) will charge for query execution.

The query returns charge for executing a particular query (identified by name) for the user whose ID code is inside query's header.

Input body

String -- query (name of method we are asking about)

Output body

Struct -- body of response

Int amount -- charge

The method has header, query and body components.
The body of method input consists of method's name. The body of output is a structure containing the field **amount**, which contains the charge applicable to this user at this moment in cents.

5.2.5 Query for loading a classifier loadClassificator

Query *dataset.loadClassificator* (where *dataset* is dataset's name) returns a list of classifiers or the contents or a subset of contents of a classifier.

Query for listing classifiers

Input body

String -- empty string

Output body

Array klassifikaatorinimed -- list of classifiers available in this dataset
    String -- name of classifier

Query for contents of a classifier

Input body

Struct -- description of data
    String klassifikaatori_nimi -- name of classifier
    String alamosa -- definition of a subset of the classifiers's contents, eg name of branch of a tree-shaped classifier
    Date alates -- start moment for changes to classifier contents
    String maksimum -- maximum limit of data to be returned

Output body

Struct -- subset of classifier's contents to be returned

If query input is empty, then a list of classifier names is returned.

A subset of a classifier is defined by the date parameter *alates*, which sets the earliest date for modifications to the classifier that we are interested in, and /or parameter *alamosa*, which defines a part of the classifier tree, eg using a LDAP identifier.

Output body's structure corresponds to the LDAP tree structure of response data. A LDAP record is presented as a structure with sub-parameters containing fields of that record and lists of sub-records. Elements of lists in sub-records are anonymous; names of lists are values of the objectClass field of the LDAP records corresponding to elements. If a field of a LDAP record has several values, then the field is presented as a list of anonymous values, and the name of the list is the name of the field.

5.2.6 Query for entering a legacy system legacyXYZ

Queries with names in the form *dataset.legacyXYZ* (where *dataset* is dataset's name and XYZ is arbitrary text characteristic to that query) are queries for entering legacy systems. This type of query allows to use the authentication and authorization means of an information system integrated with X-tee in other information systems.

Input body

Array
    String -- names of queries allowed to that person

Output body
String url -- URL of the service

The information system, when creating the query, adds all names of legacy system queries allowed for that person in given dataset to the body of the query.
The security server of the dataset checks, whether the institution is allowed to execute all the queries listed. If the list contains queries that the institution has no right to execute, then the security server will respond with fault message and will refuse to service the query.
The URL in the query response is a unique URL created by adapter server and/or legacy system's server that identifies the session uniquely. An example of the URL would look like
https://legacy.xxx.ee/cgi-bin/legacy.cgi?id=XXXXXXXXXXXXXXXXXX

6. References

[WSDL]
Web Services Description Language (WSDL) 1.1
http://www.w3.org/TR/wsdl

[XSD]
XML Schema Part 2: Datatypes
http://www.w3.org/TR/xmlschema-2/

[SOAP]
Simple Object Access Protocol (SOAP) 1.1
http://www.w3.org/TR/SOAP/

[XMLRPC]
XML-RPC Specification
http://www.xmlrpc.com/spec

[XVEAD]
XMLRPC veakoodid
xmlrpc_veateated.pdf

7. Examples

7.1 SOAP examples

7.1.1 Query listProducers

Input

```
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:ns4="http://x-tee.riik.ee/xsd/xtee.xsd"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Header>
      <ns4:asutus xsi:type="xsd:string">10239452</ns4:asutus>
      <ns4:andmekogu xsi:type="xsd:string">xtee</ns4:andmekogu>
      <ns4:ametnik xsi:type="xsd:string">370101010007</ns4:ametnik>
      <ns4:id
         xsi:type="xsd:string">990c046413466aa32a80d3c472326a112f0
         48571</ns4:id>
      <ns4:nimi xsi:type="xsd:string">xtee.listProducers</ns4:nimi>
      <ns4:toimik xsi:type="xsd:string" />
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
      <ns4:listProducers>
        <keha xsi:type="xsd:string" />
      </ns4:listProducers>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Output

```
<?xml version="1.0" encoding="utf-8" ?>
- <SOAP-ENV:Envelope xmlns:SOAP-
    ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
    ENC="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:ns4="http://x-tee.riik.ee/xsd/xtee.xsd" SOAP-
    ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  - <SOAP-ENV:Body>
    - <ns4:listProducersResponse>
      - <keha xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="SOAP-
            ENC:Struct[3]" SOAP-ENC:offset="[0]">
```

```
        - <item>
            <name xsi:type="xsd:string">autoregister</name>
            <description xsi:type="xsd:string">Eesti riiklik
              liiklusregister</description>
          </item>
        - <item>
            <name xsi:type="xsd:string">rr</name>
            <description
              xsi:type="xsd:string">Rahvastikuregister</descriptio
              n>
          </item>
        - <item>
            <name xsi:type="xsd:string">maakataster</name>
            <description
              xsi:type="xsd:string">Maakataster</description>
          </item>
        </keha>
      </ns4:listProducersResponse>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

7.1.2 Query allowedMethods

Input

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <SOAP-ENV:Envelope xmlns:SOAP-
    ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
    ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:ns4="http://x-
    tee.riik.ee/xsd/xtee.xsd" SOAP-
    ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  - <SOAP-ENV:Header>
      <ns4:asutus xsi:type="xsd:string">10239452</ns4:asutus>
      <ns4:andmekogu xsi:type="xsd:string">maakataster</ns4:andmekogu>
      <ns4:ametnik xsi:type="xsd:string">30101010007</ns4:ametnik>
      <ns4:id
        xsi:type="xsd:string">411d6755661409fed365ad8135f8210be07613da</
        ns4:id>
      <ns4:nimi xsi:type="xsd:string">maakataster.allowedMethods</ns4:nimi>
      <ns4:toimik xsi:type="xsd:string" />
    </SOAP-ENV:Header>
  - <SOAP-ENV:Body>
    - <ns4:allowedMethods>
        <keha xsi:type="xsd:string" />
```

```
      </ns4:allowedMethods>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

Output

```
  <?xml version="1.0" encoding="utf-8" ?>
- <SOAP-ENV:Envelope xmlns:SOAP-
    ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
    ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:ns4="http://x-
    tee.riik.ee/xsd/xtee.xsd" SOAP-
    ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  - <SOAP-ENV:Header xmlns:xtpais="http://x-tee.riik.ee/xsd/xtee.xsd">
      <xtpais:asutus xsi:type="xsd:string">10239452</xtpais:asutus>
      <xtpais:andmekogu xsi:type="xsd:string">maakataster</xtpais:andmekogu>
      <xtpais:ametnik xsi:type="xsd:string">30101010007</xtpais:ametnik>
      <xtpais:id
        xsi:type="xsd:string">411d6755661409fed365ad8135f8210be07613da</x
        tpais:id>
      <xtpais:nimi xsi:type="xsd:string">maakataster.allowedMethods</xtpais:nimi>
      <xtpais:toimik xsi:type="xsd:string" />
    </SOAP-ENV:Header>
  - <SOAP-ENV:Body>
    - <ns4:allowedMethodsResponse>
      - <keha xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="xsd:string[2]"
          SOAP-ENC:offset="[0]">
          <item xsi:type="xsd:string">maakataster.ky.v1</item>
          <item xsi:type="xsd:string">maakataster.ky_aadr.v1</item>
        </keha>
      </ns4:allowedMethodsResponse>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

7.1.3 Query listMethods

Input

```
  <?xml version="1.0" encoding="utf-8" ?>

- <SOAP-ENV:Envelope xmlns:SOAP-
    ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
    ENC="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:ns4="http://x-tee.riik.ee/xsd/xtee.xsd" SOAP-
    ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  - <SOAP-ENV:Body>
```

```
      <ns4:listMethods />
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

Output

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <SOAP-ENV:Envelope xmlns:SOAP-
    ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
    ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:ns4="http://x-
    tee.riik.ee/xsd/xtee.xsd" SOAP-
    ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  - <SOAP-ENV:Body>
    - <ns4:listMethodsResponse>
      - <keha xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="xsd:string[5]"
          SOAP-ENC:offset="[0]">
          <item xsi:type="xsd:string">maakataster.listMethods</item>
          <item xsi:type="xsd:string">maakataster.ky.v1</item>
          <item xsi:type="xsd:string">maakataster.ky_aadr.v1</item>
          <item xsi:type="xsd:string">maakataster.ky_kits.v1</item>
          <item xsi:type="xsd:string">maakataster.legacy1.v1</item>
        </keha>
      </ns4:listMethodsResponse>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

7.1.4 Data query

Input

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <SOAP-ENV:Envelope xmlns:SOAP-
    ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
    ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:ns4="http://x-
    tee.riik.ee/xsd/xtee.xsd"
    xmlns:ns5="http://producers.maakataster.xtee.riik.ee/producer/maakataster"
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  - <SOAP-ENV:Header>
      <ns4:asutus xsi:type="xsd:string">10239452</ns4:asutus>
      <ns4:andmekogu xsi:type="xsd:string">maakataster</ns4:andmekogu>
      <ns4:ametnik xsi:type="xsd:string">30101010007</ns4:ametnik>
      <ns4:id
        xsi:type="xsd:string">3aed1ae3813eb7fbed9396fda70ca1215d3f3fe1</ns
        4:id>
```

```
      <ns4:nimi xsi:type="xsd:string">maakataster.ky_aadr.v1</ns4:nimi>
      <ns4:toimik xsi:type="xsd:string" />
    </SOAP-ENV:Header>
  - <SOAP-ENV:Body>
    - <ns5:ky_aadr>
      - <keha>
          <maakond xsi:type="ns4:maakond">0037</maakond>
          <omavalitsus xsi:type="ns4:vald">0784</omavalitsus>
          <asustusyksus xsi:type="ns4:asula" />
          <ky_ametlik_nimetus xsi:type="xsd:string">risttee
            plats*</ky_ametlik_nimetus>
          <ky_max xsi:type="xsd:string">100</ky_max>
        </keha>
      </ns5:ky_aadr>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

Output

```
  <?xml version="1.0" encoding="utf-8" ?>
- <SOAP-ENV:Envelope xmlns:SOAP-
    ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
    ENC="http://schemas.xmlsoap.org/soap/encoding/" SOAP-
    ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  - <SOAP-ENV:Header xmlns:xtpais="http://x-tee.riik.ee/xsd/xtee.xsd">
      <xtpais:asutus xsi:type="xsd:string">10239452</xtpais:asutus>
      <xtpais:andmekogu xsi:type="xsd:string">maakataster</xtpais:andmekogu>
      <xtpais:ametnik xsi:type="xsd:string">30101010007</xtpais:ametnik>
      <xtpais:id
        xsi:type="xsd:string">3aed1ae3813eb7fbed9396fda70ca1215d3f3fe1</xtp
        ais:id>
      <xtpais:nimi xsi:type="xsd:string">maakataster.ky_aadr.v1</xtpais:nimi>
      <xtpais:toimik xsi:type="xsd:string" />
    </SOAP-ENV:Header>
  - <SOAP-ENV:Body>
    - <ns4:ky_aadrResponse
        xmlns:ns4="http://producers.maakataster.xtee.riik.ee/producer/maakata
        ster">
      - <paring>
          <maakond xsi:type="xsd:string">0037</maakond>
          <omavalitsus xsi:type="xsd:string">0784</omavalitsus>
          <asustusyksus xsi:type="xsd:string" />
          <ky_ametlik_nimetus xsi:type="xsd:string">risttee
            plats*</ky_ametlik_nimetus>
          <ky_max xsi:type="xsd:string">100</ky_max>
        </paring>
      - <keha xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="SOAP-
```

```
            ENC:Struct[3]" SOAP-ENC:offset="[0]">
          - <item>
              <katastritunnus xsi:type="xsd:string">012:345:67</katastritunnus>
              <ky_asukoht xsi:type="xsd:string">Risttee plats 1A</ky_asukoht>
              <maakond xsi:type="xsd:string">0037</maakond>
              <omavalitsus xsi:type="xsd:string">0784</omavalitsus>
              <registreeritud xsi:type="xsd:dateTime">1999-09-
                19T19:09:10</registreeritud>
              <sihtotstarve1 xsi:type="xsd:string">001</sihtotstarve1>
            </item>
          - <item>
              <katastritunnus xsi:type="xsd:string">012:345:68</katastritunnus>
              <ky_asukoht xsi:type="xsd:string">Risttee plats 1B</ky_asukoht>
              <maakond xsi:type="xsd:string">0037</maakond>
              <omavalitsus xsi:type="xsd:string">0784</omavalitsus>
              <registreeritud xsi:type="xsd:dateTime">1999-09-
                19T19:09:10</registreeritud>
              <sihtotstarve1 xsi:type="xsd:string">001</sihtotstarve1>
            </item>
          </keha>
        </ns4:ky_aadrResponse>
      </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>
```

7.1.5 Query andmekogu.loadClassificator

Input to query for listing classifiers

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <SOAP-ENV:Envelope xmlns:SOAP-
    ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
    ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:ns4="http://x-
    tee.riik.ee/xsd/xtee.xsd" SOAP-
    ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  - <SOAP-ENV:Header>
      <ns4:asutus xsi:type="xsd:string">10239452</ns4:asutus>
      <ns4:andmekogu xsi:type="xsd:string">assert</ns4:andmekogu>
      <ns4:ametnik xsi:type="xsd:string">30101010007</ns4:ametnik>
      <ns4:id
        xsi:type="xsd:string">3e47be25da43528deb639f5965d58b4d21bbc173</n
        s4:id>
      <ns4:nimi xsi:type="xsd:string">assert.loadClassificator</ns4:nimi>
      <ns4:toimik xsi:type="xsd:string" />
    </SOAP-ENV:Header>
  - <SOAP-ENV:Body>
    - <ns4:loadClassificator>
```

```
        <keha xsi:type="xsd:string" />
      </ns4:loadClassificator>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

Output with the list of classifiers

```
<?xml version="1.0" encoding="UTF-8" ?>
- <SOAP-ENV:Envelope xmlns:SOAP-
    ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
    ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:ns4="http://x-
    tee.riik.ee/xsd/xtee.xsd" xmlns:ns5="http://x-tee.riik.ee/xsd/xtee.xsd" SOAP-
    ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  - <SOAP-ENV:Header>
      <ns4:asutus xsi:type="xsd:string">10239452</ns4:asutus>
      <ns4:andmekogu xsi:type="xsd:string">assert</ns4:andmekogu>
      <ns4:ametnik xsi:type="xsd:string">30101010007</ns4:ametnik>
      <ns4:id
        xsi:type="xsd:string">3e47be25da43528deb639f5965d58b4d21bbc173</n
        s4:id>
      <ns4:nimi xsi:type="xsd:string">assert.loadClassificator</ns4:nimi>
      <ns4:toimik xsi:type="xsd:string" />
    </SOAP-ENV:Header>
  - <SOAP-ENV:Body>
    - <ns5:loadClassificatorResponse>
        <paring xsi:type="xsd:string" />
      - <keha>
        - <klassifikaatorinimed xsi:type="SOAP-ENC:Array" SOAP-
            ENC:arrayType="xsd:string[2]" SOAP-ENC:offset="[0]">
            <item xsi:type="xsd:string">EHAK</item>
            <item xsi:type="xsd:string">abc</item>
          </klassifikaatorinimed>
        </keha>
      </ns5:loadClassificatorResponse>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

Input to query for loading a classifier

```
<?xml version="1.0" encoding="UTF-8" ?>
- <SOAP-ENV:Envelope xmlns:SOAP-
    ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

```
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
    ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:ns4="http://x-
    tee.riik.ee/xsd/xtee.xsd" xmlns:ns5="http://x-tee.riik.ee/xsd/xtee.xsd" SOAP-
    ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  - <SOAP-ENV:Header>
      <ns4:asutus xsi:type="xsd:string">10239452</ns4:asutus>
      <ns4:andmekogu xsi:type="xsd:string">assert</ns4:andmekogu>
      <ns4:ametnik xsi:type="xsd:string">30101010007</ns4:ametnik>
      <ns4:id
        xsi:type="xsd:string">00f340cc5da4d3102c0859cc5f2ae2547b07b2c3</ns
        4:id>
      <ns4:nimi xsi:type="xsd:string">assert.loadClassificator</ns4:nimi>
      <ns4:toimik xsi:type="xsd:string" />
    </SOAP-ENV:Header>
  - <SOAP-ENV:Body>
    - <ns5:loadClassificator>
      - <keha>
          <klassifikaatori_nimi xsi:type="xsd:string">EHAK</klassifikaatori_nimi>
          <alamosa />
          <alates />
          <maksimum />
        </keha>
      </ns5:loadClassificator>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

Output with the contents of the classifier

```
<?xml version="1.0" encoding="UTF-8" ?>
- <SOAP-ENV:Envelope xmlns:SOAP-
    ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
    ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:ns4="http://x-
    tee.riik.ee/xsd/xtee.xsd" xmlns:ns5="http://x-tee.riik.ee/xsd/xtee.xsd" SOAP-
    ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  - <SOAP-ENV:Header>
      <ns4:asutus xsi:type="xsd:string">10239452</ns4:asutus>
      <ns4:andmekogu xsi:type="xsd:string">assert</ns4:andmekogu>
      <ns4:ametnik xsi:type="xsd:string">30101010007</ns4:ametnik>
      <ns4:id
        xsi:type="xsd:string">00f340cc5da4d3102c0859cc5f2ae2547b07b2c3</ns
        4:id>
      <ns4:nimi xsi:type="xsd:string">assert.loadClassificator</ns4:nimi>
      <ns4:toimik xsi:type="xsd:string" />
    </SOAP-ENV:Header>
  - <SOAP-ENV:Body>
    - <ns5:loadClassificatorResponse>
      - <paring>
```

```
            <klassifikaatori_nimi xsi:type="xsd:string">EHAK</klassifikaatori_nimi>
            <alamosa xsi:type="xsd:string" />
            <alates xsi:type="xsd:string" />
            <maksimum xsi:type="xsd:string" />
          </paring>
        - <keha>
          - <klassifikaatorid xsi:type="SOAP-ENC:Array" SOAP-
              ENC:arrayType="Struct[2]" SOAP-ENC:offset="[0]">
            - <item>
                <l xsi:type="xsd:string">37</l>
                <objectclass xsi:type="xsd:string">xteeLocality</objectclass>
                <cn xsi:type="xsd:string">Harju</cn>
                <ehakType xsi:type="xsd:string">0</ehakType>
                <description xsi:type="xsd:string">maakond</description>
              - <alamklassifikaatorid xsi:type="SOAP-ENC:Array" SOAP-
                  ENC:arrayType="Struct[1]" SOAP-ENC:offset="[0]">
                - <item>
                    <l xsi:type="xsd:string">112</l>
                    <objectclass xsi:type="xsd:string">xteeLocality</objectclass>
                    <cn xsi:type="xsd:string">Aegviidu</cn>
                    <ehakType xsi:type="xsd:string">1</ehakType>
                    <description xsi:type="xsd:string">vald</description>
                  </item>
                </alamklassifikaatorid>
              </item>
            - <item>
                <l xsi:type="xsd:string">39</l>
                <objectclass xsi:type="xsd:string">xteeLocality</objectclass>
                <cn xsi:type="xsd:string">Hiiu</cn>
                <ehakType xsi:type="xsd:string">0</ehakType>
                <description xsi:type="xsd:string">maakond</description>
              </item>
            </klassifikaatorid>
            <klassifikaatorinimed xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="[0]"
              />
          </keha>
        </ns5:loadClassificatorResponse>
      </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>
```

7.2 XMLRPC examples

7.2.1 Query listProducers

`Input`

```
<?xml version="1.0" encoding="UTF-8" ?>
```

```
<methodCall>
    <methodName>xtee.listProducers</methodName>
   <params>
     <param>
       <value>
          <string />
       </value>
     </param>
   </params>
 </methodCall>
```

Output

```
<?xml version="1.0" encoding="utf-8" ?>
- <methodResponse>
  - <params>
    - <param>
      - <value>
        - <array>
          - <data>
            - <value>
              - <struct>
                - <member>
                    <name>name</name>
                  - <value>
                      <string>autoregister</string>
                    </value>
                  </member>
                - <member>
                    <name>description</name>
                  - <value>
                      <string>Eesti riiklik autoregister</string>
                    </value>
                  </member>
                </struct>
              </value>
            - <value>
              - <struct>
                - <member>
                    <name>name</name>
                  - <value>
                      <string>rr</string>
                    </value>
                  </member>
                - <member>
                    <name>description</name>
                  - <value>
                      <string>Rahvastikuregister</string>
```

```
                    </value>
                  </member>
                </struct>
              </value>
            - <value>
              - <struct>
                - <member>
                    <name>name</name>
                  - <value>
                      <string>maakataster</string>
                    </value>
                  </member>
                - <member>
                    <name>description</name>
                  - <value>
                      <string>Maakataster</string>
                    </value>
                  </member>
                </struct>
              </value>
            </data>
          </array>
        </value>
      </param>
    </params>
  </methodResponse>
```

7.2.2 Query allowedMethods

Input

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <methodCall>
    <methodName>maakataster.allowedMethods</methodName>
  - <params>
    - <param>
      - <value>
        - <struct>
          - <member>
              <name>asutus</name>
            - <value>
                <string>10239452</string>
              </value>
            </member>
          - <member>
              <name>andmekogu</name>
            - <value>
                <string>maakataster</string>
              </value>
```

```
            </member>
          - <member>
              <name>ametnik</name>
            - <value>
                <string>30101010007</string>
              </value>
            </member>
          - <member>
              <name>id</name>
            - <value>
                <string>735c380d1c319482355c2a28977fa4cc0ada041b</string>
              </value>
            </member>
          - <member>
              <name>nimi</name>
            - <value>
                <string>maakataster.allowedMethods</string>
              </value>
            </member>
          - <member>
              <name>toimik</name>
            - <value>
                <string />
              </value>
            </member>
          </struct>
        </value>
      </param>
    - <param>
      - <value>
          <string />
        </value>
      </param>
    </params>
  </methodCall>
```

Output

```
<?xml version="1.0" encoding="utf-8" ?>
- <methodResponse>
  - <params>
    - <param>
      - <value>
        - <struct>
          - <member>
              <name>pais</name>
            - <value>
```

```
- <struct>
  - <member>
      <name>asutus</name>
    - <value>
        <string>10239452</string>
      </value>
    </member>
  - <member>
      <name>andmekogu</name>
    - <value>
        <string>maakataster</string>
      </value>
    </member>
  - <member>
      <name>ametnik</name>
    - <value>
        <string>30101010007</string>
      </value>
    </member>
  - <member>
      <name>id</name>
    - <value>

          <string>735c380d1c319482355c2a28977fa4cc0ada041b
          </string>
      </value>
    </member>
  - <member>
      <name>nimi</name>
    - <value>
        <string>maakataster.allowedMethods</string>
      </value>
    </member>
  - <member>
      <name>toimik</name>
    - <value>
        <string />
      </value>
    </member>
  </struct>
</value>
</member>
- <member>
  <name>paring</name>
- <value>
    <string />
  </value>
</member>
```

```
        - <member>
            <name>keha</name>
          - <value>
            - <array>
              - <data>
                - <value>
                    <string>maakataster.ky.v1</string>
                  </value>
                - <value>
                    <string>maakataster.ky_aadr.v1</string>
                  </value>
                </data>
              </array>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodResponse>
```

7.2.3 Query listMethods

Input

```
<?xml version="1.0" encoding="UTF-8" ?>
- <methodCall>
    <methodName>system.listMethods</methodName>
    <params />
  </methodCall>
```

Output

```
<?xml version="1.0" encoding="UTF-8" ?>
- <methodResponse>
  - <params>
    - <param>
      - <value>
        - <array>
          - <data>
            - <value>
                <string>system.listMethods</string>
              </value>
            - <value>
                <string>maakataster.ky.v1</string>
              </value>
```

```
            - <value>
                <string>maakataster.ky_aadr.v1</string>
              </value>
            - <value>
                <string>maakataster.ky_kits.v1</string>
              </value>
            - <value>
                <string>maakataster.legacy1.v1</string>
              </value>
            </data>
          </array>
        </value>
      </param>
    </params>
  </methodResponse>
```

7.2.4 Data query

`Input`

```
<?xml version="1.0" encoding="UTF-8" ?>
- <methodCall>
    <methodName>maakataster.ky_aadr.v1</methodName>
  - <params>
    - <param>
      - <value>
        - <struct>
          - <member>
              <name>asutus</name>
            - <value>
                <string>10239452</string>
              </value>
            </member>
          - <member>
              <name>andmekogu</name>
            - <value>
                <string>maakataster</string>
              </value>
            </member>
          - <member>
              <name>ametnik</name>
            - <value>
                <string>30101010007</string>
              </value>
            </member>
          - <member>
              <name>id</name>
            - <value>
```

```
            <string>57bbbf4410f9fede5f4e6b3b94dcbaf48865541f</string>
          </value>
        </member>
      - <member>
          <name>nimi</name>
        - <value>
            <string>maakataster.ky_aadr.v1</string>
          </value>
        </member>
      - <member>
          <name>toimik</name>
        - <value>
            <string />
          </value>
        </member>
      </struct>
    </value>
  </param>
- <param>
  - <value>
    - <struct>
      - <member>
          <name>maakond</name>
        - <value>
            <string>0037</string>
          </value>
        </member>
      - <member>
          <name>omavalitsus</name>
        - <value>
            <string>0784</string>
          </value>
        </member>
      - <member>
          <name>asustusyksus</name>
        - <value>
            <string />
          </value>
        </member>
      - <member>
          <name>ky_ametlik_nimetus</name>
        - <value>
            <string>risttee plats*</string>
          </value>
        </member>
      - <member>
          <name>ky_max</name>
        - <value>
```

```
                <string>100</string>
              </value>
            </member>
          </struct>
        </value>
      </param>
    </params>
  </methodCall>
```

Output

```
  <?xml version="1.0" encoding="utf-8" ?>
- <methodResponse>
  - <params>
    - <param>
      - <value>
        - <struct>
          - <member>
              <name>pais</name>
            - <value>
              - <struct>
                - <member>
                    <name>asutus</name>
                  - <value>
                      <string>10239452</string>
                    </value>
                  </member>
                - <member>
                    <name>andmekogu</name>
                  - <value>
                      <string>maakataster</string>
                    </value>
                  </member>
                - <member>
                    <name>ametnik</name>
                  - <value>
                      <string>30101010007</string>
                    </value>
                  </member>
                - <member>
                    <name>id</name>
                  - <value>

                      <string>57bbbf4410f9fede5f4e6b3b94dcbaf48865541f</
                      string>
                    </value>
                  </member>
                - <member>
```

```
            <name>nimi</name>
          - <value>
              <string>maakataster.ky_aadr.v1</string>
            </value>
          </member>
        - <member>
            <name>toimik</name>
          - <value>
              <string />
            </value>
          </member>
        </struct>
      </value>
    </member>
  - <member>
      <name>paring</name>
    - <value>
      - <struct>
        - <member>
            <name>maakond</name>
          - <value>
              <string>0037</string>
            </value>
          </member>
        - <member>
            <name>omavalitsus</name>
          - <value>
              <string>0784</string>
            </value>
          </member>
        - <member>
            <name>asustusyksus</name>
          - <value>
              <string />
            </value>
          </member>
        - <member>
            <name>ky_ametlik_nimetus</name>
          - <value>
              <string>risttee plats*</string>
            </value>
          </member>
        - <member>
            <name>ky_max</name>
          - <value>
              <string>100</string>
            </value>
          </member>
```

```
        </struct>
      </value>
    </member>
  - <member>
      <name>keha</name>
    - <value>
      - <array>
        - <data>
          - <value>
            - <struct>
              - <member>
                  <name>katastritunnus</name>
                - <value>
                    <string>012:345:67</string>
                  </value>
                </member>
              - <member>
                  <name>ky_ametlik_nimetus</name>
                - <value>
                    <string>Risttee plats 1A</string>
                  </value>
                </member>
              - <member>
                  <name>registreeritud</name>
                - <value>

                      <dateTime.iso8601>19990919T19:09:10</dateTime.i
                      so8601>
                  </value>
                </member>
              - <member>
                  <name>sihtotstarve1</name>
                - <value>
                    <string>001</string>
                  </value>
                </member>
              </struct>
            </value>
          - <value>
            - <struct>
              - <member>
                  <name>katastritunnus</name>
                - <value>
                    <string>012:345:68</string>
                  </value>
                </member>
              - <member>
                  <name>ky_ametlik_nimetus</name>
```

```
                    - <value>
                        <string>Risttee plats 1B</string>
                      </value>
                    </member>
                  - <member>
                      <name>registreeritud</name>
                    - <value>

                          <dateTime.iso8601>19990919T19:09:10</dateTime.i
                          so8601>
                      </value>
                    </member>
                  - <member>
                      <name>sihtotstarve1</name>
                    - <value>
                        <string>001</string>
                      </value>
                    </member>
                  </struct>
                </value>
              </data>
            </array>
          </value>
        </member>
      </struct>
    </value>
  </param>
 </params>
</methodResponse>
```

7.3 WSDL example

```
<?xml version="1.0" ?>
- <definitions name="mydef"
    targetNamespace="http://producers.maakataster.xtee.riik.ee/producer/maakat
    aster"
    xmlns:tns="http://producers.maakataster.xtee.riik.ee/producer/maakataster"
    xmlns:xtee="http://x-tee.riik.ee/xsd/xtee.xsd"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xlink="http://www.w3.org/1999/xlink"
   xmlns:SOAP-ENC="http://schemas.xmlsoap.org/wsdl/encoding/"
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/wsdl/envelope/">
  - <types>
    - <schema
        targetNamespace="http://producers.maakataster.xtee.riik.ee/producer/ma
        akataster" xmlns="http://www.w3.org/2001/XMLSchema">
```

```
- <element name="ky_aadr_paring">
  - <complexType>
    - <all>
      - <element name="maakond" type="xtee:maakond">
        - <annotation>
          - <appinfo>
              <xtee:title>Maakonna kood</xtee:title>
            </appinfo>
          </annotation>
        </element>
      - <element name="omavalitsus" type="xtee:vald">
        - <annotation>
          - <appinfo>
              <xtee:ref>maakond</xtee:ref>
              <xtee:title>Omavalitsuse kood</xtee:title>
            </appinfo>
          </annotation>
        </element>
      - <element name="asustusyksus" minOccurs="0" type="xtee:asula">
        - <annotation>
          - <appinfo>
              <xtee:ref>omavalitsus</xtee:ref>
              <xtee:title>Asustusüksuse kood</xtee:title>
            </appinfo>
          </annotation>
        </element>
      - <element name="ky_ametlik_nimetus" minOccurs="0" type="string">
        - <annotation>
          - <appinfo>
              <xtee:wildcard>*?</xtee:wildcard>
              <xtee:title>Katastriüksuse ametlik nimetus</xtee:title>
            </appinfo>
          </annotation>
        </element>
      - <element name="ky_max" minOccurs="0" type="tns:t_ky_max">
        - <annotation>
          - <appinfo>
              <xtee:title>Vastuste maksimaalne arv. Vaikimisi 10</xtee:title>
            </appinfo>
          </annotation>
        </element>
      </all>
    </complexType>
  </element>
- <element name="ky_paring">
  - <complexType>
    - <all>
      - <element name="katastritunnus" type="string">
```

```
            - <annotation>
              - <appinfo>
                  <xtee:wildcard>*?</xtee:wildcard>
                  <xtee:title>Katastriüksuse tunnus</xtee:title>
                </appinfo>
              </annotation>
            </element>
          - <element name="ky_max" minOccurs="0" type="tns:t_ky_max">
            - <annotation>
              - <appinfo>
                  <xtee:title>Vastuste maksimaalne arv. Vaikimisi 10</xtee:title>
                </appinfo>
              </annotation>
            </element>
          </all>
        </complexType>
      </element>
    - <element name="ky_vastus" type="SOAP-ENC:Array">
      - <complexType>
        - <all>
          - <element minOccurs="0" maxOccurs="unbounded">
            - <annotation>
              - <appinfo>
                  <xtee:title>Katastriüksuse andmed</xtee:title>
                </appinfo>
              </annotation>
            - <complexType>
              - <all>
                - <element name="katastritunnus" type="string">
                  - <annotation>
                    - <appinfo>
                        <xtee:title>Katastritunnus</xtee:title>
                      </appinfo>
                    </annotation>
                  </element>
                - <element name="ky_ametlik_nimetus" minOccurs="0"
                    type="string">
                  - <annotation>
                    - <appinfo>
                        <xtee:title>Katastriüksuse ametlik nimetus</xtee:title>
                      </appinfo>
                    </annotation>
                  </element>
                - <element name="registreeritud" type="dateTime">
                  - <annotation>
                    - <appinfo>
                        <xtee:title>Registreerimise kuupäev</xtee:title>
                      </appinfo>
```

```
                    </annotation>
                  </element>
                - <element name="sihtotstarve1" minOccurs="0"
                    type="tns:t_sihtotstarve">
                  - <annotation>
                    - <appinfo>
                        <xtee:title>Esimese sihtotstarbe tehniline
                          kood</xtee:title>
                      </appinfo>
                    </annotation>
                  </element>
                </all>
              </complexType>
            </element>
          </all>
        </complexType>
      </element>
    - <element name="ky_kits_paring">
      - <complexType>
        - <all>
          - <element name="katastritunnus" type="string">
            - <annotation>
              - <appinfo>
                  <xtee:title>Katastritunnuste loetelu</xtee:title>
                </appinfo>
              </annotation>
            </element>
          </all>
        </complexType>
      </element>
    - <element name="ky_kits_vastus" type="SOAP-ENC:Array">
      - <complexType>
        - <all>
          - <element minOccurs="0" maxOccurs="unbounded">
            - <annotation>
              - <appinfo>
                  <xtee:title>Katastriüksust kitsendav objekt</xtee:title>
                </appinfo>
              </annotation>
            - <complexType>
              - <all>
                - <element name="katastritunnus" type="string">
                  - <annotation>
                    - <appinfo>
                        <xtee:title>Katastriüksuse tunnus</xtee:title>
                      </appinfo>
                    </annotation>
                  </element>
```

```
                - <element name="nahtus" type="string">
                  - <annotation>
                    - <appinfo>
                        <xtee:title>Kitsendust põhjustav nähtus</xtee:title>
                      </appinfo>
                    </annotation>
                  </element>
                - <element name="pindala" minOccurs="0" type="double">
                  - <annotation>
                    - <appinfo>
                        <xtee:title>Kitsenduse pindala ruutmeetrites</xtee:title>
                      </appinfo>
                    </annotation>
                  </element>
                </all>
              </complexType>
            </element>
          </all>
        </complexType>
      </element>
    - <simpleType name="t_sihtotstarve">
      - <annotation>
        - <appinfo>
            <xtee:title>Sihtotstarbe tüüp</xtee:title>
          </appinfo>
        </annotation>
      - <restriction base="string">
        - <enumeration value="001">
          - <annotation>
            - <appinfo>
                <xtee:title>(1)Elamumaa (E)</xtee:title>
              </appinfo>
            </annotation>
          </enumeration>
        - <enumeration value="0010">
          - <annotation>
            - <appinfo>
                <xtee:title>(010) Kaitsealune maa (H)</xtee:title>
              </appinfo>
            </annotation>
          </enumeration>
        </restriction>
      </simpleType>
    - <simpleType name="t_ky_max">
      - <annotation>
        - <appinfo>
            <xtee:title>Vastuste maksimaalne arv</xtee:title>
          </appinfo>
```

```
            </annotation>
          - <restriction base="string">
            - <enumeration value="10">
              - <annotation>
                - <appinfo>
                    <xtee:title>10</xtee:title>
                  </appinfo>
                </annotation>
              </enumeration>
            - <enumeration value="100">
              - <annotation>
                - <appinfo>
                    <xtee:title>100</xtee:title>
                  </appinfo>
                </annotation>
              </enumeration>
            </restriction>
          </simpleType>
        </schema>
      </types>
    - <binding name="mybinding" type="myporttype">
      - <operation name="ky_aadr">
          <xtee:version>v1</xtee:version>
        </operation>
      - <operation name="ky">
          <xtee:version>v1</xtee:version>
        </operation>
      - <operation name="ky_kits">
          <xtee:version>v1</xtee:version>
        </operation>
      - <operation name="legacy1">
          <xtee:version>v1</xtee:version>
        </operation>
      </binding>
    - <portType name="myporttype">
      - <operation name="ky_aadr">
          <input message="tns:ky_aadr" />
          <output message="tns:ky_aadrResponse" />
        - <documentation>
            <xtee:title>Katastriüksuse päring asukoha järgi</xtee:title>
          </documentation>
        </operation>
      - <operation name="ky">
          <input message="tns:ky" />
          <output message="tns:kyResponse" />
        - <documentation>
            <xtee:title xml:lang="ee">Katastriüksuse päring tunnuse järgi</xtee:title>
            <xtee:title xml:lang="pt">Inquira por código</xtee:title>
```

```
            <xtee:title xml:lang="en">Query by code</xtee:title>
            <xtee:notes>Päring väljastab selle osa andmetest, mis on registrisse
              kantud</xtee:notes>
          </documentation>
        </operation>
      - <operation name="ky_kits">
          <input message="tns:ky_kits" />
          <output message="tns:ky_kitsResponse" />
        - <documentation>
            <xtee:title>Katastriüksust kitsendavad objektid</xtee:title>
          </documentation>
        </operation>
      - <operation name="legacy1">
        - <documentation>
            <xtee:title>Sisenemine ühte hoopis teise
              infosüsteemi</xtee:title>
          </documentation>
          <input message="tns:legacy" />
          <output message="tns:legacyResponse" />
        </operation>
      </portType>
    - <message name="ky_aadr">
        <part name="keha" element="tns:ky_aadr_paring" />
      </message>
    - <message name="ky_aadrResponse">
        <part name="paring" element="tns:ky_aadr_paring" />
        <part name="keha" element="tns:ky_vastus" />
      </message>
    - <message name="ky">
        <part name="keha" element="tns:ky_paring" />
      </message>
    - <message name="kyResponse">
        <part name="paring" element="tns:ky_paring" />
        <part name="keha" element="tns:ky_vastus" />
      </message>
    - <message name="ky_kits">
        <part name="keha" element="tns:ky_kits_paring" />
      </message>
    - <message name="ky_kitsResponse">
        <part name="paring" element="tns:ky_kits_paring" />
        <part name="keha" element="tns:ky_kits_vastus" />
      </message>
    - <message name="legacy">
        <part name="keha" type="xtee:ArrayOfString" />
      </message>
    - <message name="legacyResponse">
        <part name="paring" type="xtee:ArrayOfString" />
        <part name="keha" element="xtee:legacy_response" />
```

```
    </message>
  - <service>
    - <port name="myport" binding="mybinding">
        <xtee:title>Maakataster</xtee:title>
        <xtee:title xml:lang="en">Land Registry</xtee:title>
        <xtee:address producer="maakataster" />
      </port>
    </service>
</definitions>
```

e-System for Real Time Democratic Land-Use Planning of Urban Environment – Pilot action in Narva Community

Supported by the *LIFE program* of the European Union (*LIFE02 ENV/EE/000426*)

Project introduction

In 2002, a LIFE - environment project was started. It is called: e-System for Real Time Democratic Land-Use Planning of Urban Environment – Pilot action in Narva Community. The acronym for the project is eCommunity. The project was initiated by the Narva City Government with the help of several partners.

Location of Narva in Eastern Baltic Sea area

The core focus of the eCommunity project is to develop a city planning and environmental management system working on the internet, which is based on innovative information and communication technologies (ICT).

The main objectives of the project are:

- To promote sustainable and democratic urban planning by using the possibilities of ICT
- To promote a concept of *e-democracy* by enabling exchange of opinions and information
- To raise public awareness and involvement in planning processes
- To create a system as a tool for urban planning

The work will be carried out over a 36-month period. To accomplish the expected objectives of the project the work is divided into different work packages:

- Management
- Creating the security and process modules
- Creating the user interface
- Creating the database
- Integration of the system
- System launch in Narva
- Dissemination
- The system launch on society level

The milestones consist of different workshops (involving stakeholders, experts, and officials), creating different modules for the system, different internal deliverables for the project team and from foreseen reports.

The expected results of the project are: a well and simply usable planning system on the internet and, of course, a good dissemination campaign that will lead to informed stakeholders who are aware of the system and are capable to use it.

The eCommunity project promotes more transparency in decision-making in Narva town and therefore offers a window of opportunity to the Narva municipality to develop a more efficient dialogue with the town inhabitants and, thus, a better understanding by the public of decisions taken by the municipality.

The creation of the eCommunity web-based information system implies the development of a kind of "virtual reality" where the town spatial plans - drawn by specialists - will have their own 3D spaces, realistic models of the planned area, which can virtually be walked through and where citizens will have the possibility to make additions and corrections to the plans. These changes are automatically transformed into a new 3D model, and the responsible official will, then, only have to decide if the correction is going to be implemented or not. This innovative IT tool will allows synthesizing of concrete decisions from statements made by citizens, giving the decision-making process a new democratic identity in real time.

The system available through the regular Internet system will be developed tailor made to the needs of specific stakeholder groups and will include components such as an “investor web” or a “tourist web” site. The system will include a wide range of thematic information on sub-systems, such as a city master plan, a thematic plan of bicycle traffic, a public transport agenda, a thematic park of the old town, etc.

There are two important ideas we keep in mind when developing the system:

The amount of information is unlimited in the eCommunity system

The system can easily be implemented in any city in the EU

Vahur Sikaste, Hendrikson & KO, Project Coordinator

Why eCommunity?

The problem

Spatial decision-making is an important task for local authorities. Good or bad public decisions directly affect the management of environment and resources that, in turn, affect the quality of life of each citizen in the community.

An exploratory case study was conducted in Narva community in the North East of Estonia in 2001. After an analysis, 5 main reasons for bad public decisions were identified:

- Spatial information is always out-of-date. The collection, analysis and presentation of data take time and other resources. Thus, lack of current information obliges decision-makers to decide on the basis of out-of-date data.
- It is hard to extract useful information for opinion taking. Most of the information around municipality officials and citizens is useless. Useful information is "dissolved" there.
- Working with spatial information is hard and boring. It demands good knowledge in geography and cartography, powerful hardware and professional search of information. Thus, most citizens either avoid planning discussions or fail to participate in them.
- It is hard to continuously collect real-time public opinion. Via elections, citizens can

express their opinion in only one question only once in several years. The current web-forums cannot have legislative power because they do not give the citizens an equal number of votes. Furthermore, current web forums do not offer spatial discussions. Thus, spatial decisions still ignore the public opinion.

- It is hard to synthesize a democratic decision from a variety of spatial public opinions. If citizens expressed their opinions on a virtual map then simple vote counter could not reveal a final decision. To summarize spatial public opinions, a smart system is needed.

Goals

The objective of *eCommunity* is to promote democratic development by using opportunities offered by information technology and the WWW. The aim is to create innovative web-based software solutions, which will promote the concept of e-democracy by enabling the exchange of opinions and information. That will help raising public awareness. The aim is to produce results that can be used in planning and policymaking processes at a local level. Stronger and more efficient communities can compose their comprehensive development and site plans without help from private companies. The expected results of the project provide a whole new branch of products for all stakeholders familiar with strategic management, planning and information technology.

Here is the list of these 5 goals of the project that will solve 5 main problems.

All new spatial information is in a single system. Databases that are constantly updated communicate well with each other. The database also contains future visions such as spatial plans. Each official and even each citizen participates in updating the database.

The system provides the user with all the collected, concentrated and integrated information according to his needs. The user has to identify his/her needs and interests. The intelligent system finds the requested information from the database, processes and analyses it as required.

Spatial information is presented in user-friendly and attractive virtual reality. The user interface displays the information as requested by the user, including texts, tables, maps and 3D simulations – realistic models of a planned area. A citizen may virtually walk through a future city and make changes at the same time, thus participating in the sustainable development of the local community.

The system collects opinions of citizens via Internet in real-time according to the rules of democracy. Digital certificates give one vote to each citizen for each question. Depending on their preferences, citizens may vote either in virtual city or in any other way. Approximately 100% Internet penetration and free access avoids digital divide and enables the participation of each citizen.

Our Decision Making Module synthesizes concrete democratic decisions from a variety of public opinions, giving the decision-making process a new democratic identity in real time.

Kristjan Piirimäe, Hendrikson & KO

Participatory planning for sustainable urban development.

As urban planning evolves into the twenty-first century, the concept of public participation - in the light of concern of environment and sustainable development trends- has become an extremely important technique in promoting more representative and interactive planning decisions. In its various forms, public involvement in planning processes has so far often been instigated and bounded by the concrete context of legislative issues. The issues relating to citizen involvement and responsible participation methods by institutions are often translated into a

language (such as technical jargon used by professionals or cryptic two-dimensional drawings) that is ineffective in communicating a designer's idea, the impact of a development, or the realistic concerns of a local resident. Often, institutions use citizen participation methods based on their own motives and effectiveness, like manipulation and directive therapy where it is assumed that an action has public support simply because of the lack of substantial opposition, no real effort is made to inform the public objectively and the underlying power lies within the professional to make the decisions. Such an "instrumental" and predetermined planning philosophy perpetuates power and control of the elite rather than fosters dialogue and community consensus.

A basic understanding of interactive planning processes can be reshaped and redefined from instrumental to communicative planning through creating a medium that uses a common language in planning commission meetings, citizen review boards, or grass-roots driven community design workshops. This common language must foster the ideas that can be realized immediately in a compelling, easily understandable, and interactive environment.

Since the early nineties, a new possibility has emerged that brings the sophisticated language of planning and design into a common visual language. Internet visualization solutions in interactive planning possibilities have the potential to help shaping a new paradigm by which people are informed about and communicate issues on form, space, and quality of life. If internet based solutions are used appropriately, they can revolutionize the process of citizen participation in planning in an effective way.

Much like the common skill of interpreting realism through television, the initial idea of the ongoing project is the creation of a three-dimensional computer model of an existing environment in a real-time modeling software package that is easily understandable for everyone.

Citizens, who are usually turned off by the static and seemingly "finished" quality of an artist rendering or physical model, will more likely feel informed and empowered at first sight by a non-static, changeable visualization that stunningly shows the proposed changes to their neighborhood. Citizens often do not feel compelled to participate simply because they don't fully understand how a certain planning proposal will affect their lives in direct ways, don't have enough time to do the research, and simply don't feel equipped to make decisions.

The idea of having to learn a lot to interpret a design or understand the complexities of a planning proposal is what often discourages citizens from participating in the decision-making process. Upon the realization that one will have access to such an intuitive technique to make informed decisions, the barriers between institutions and citizens will begin to break down. Participation will become a less feared task, and the invitation becomes more open for citizens to vote for a proposal or to contact the local planning representative or even to participate in a public workshop. Thus, the communication of the technology's availability is vital.

When this tool is used, citizens are no longer limited to being a passive audience that simply sees a proposed design in all its beauty but they become active participants. During the planning process, citizens and community leaders alike will be able to work interactively in front of a computer. By simple movements of the mouse, they will be able to walk up to their own front porch, or look at the proposed shopping mall alternatives across the street from virtually any angle. The changes from the discussion can be viewed interactively again and design issues can be revisited or introduced by the participants. Participating can ultimately fuel ideas, bringing them to life and fostering clearer communication for consensus building.

Peep Leppik, Hendrikson & KO

What Narva residents want from interactive town planning – experience of the eCommunity project

The town of Narva with a population of 70 000 is situated in the northeast of Estonia; it is the third largest town in Estonia. Narva is located on the border to Russia: the frontier goes along the Narva River separating Narva from the neighboring Russian municipality of Ivangorod. The Narva municipality faces difficulties connected with the restructuring of its economy and has to address diverse socio-economic issues in this border region. The community tried to come up with fresh and innovative ideas, which could possibly help to solve existing social and economic difficulties. The eCommunity project is one of the Narva municipality international innovative projects; the EU LIFE program provided support for this project.

Peipsi Center for Transboundary Cooperation (Peipsi CTC) was invited to join the project in order to identify stakeholders and their interests in using the web-based interactive system regarding Narva town planning. Another intention was to elaborate recommendations involving the stakeholders in the town planning policy process. To identify the major stakeholder groups and their interests for this project, Peipsi CTC conducted a review of relevant documents and carried out a series of individual and group interviews with representatives of the identified major stakeholder groups in Narva municipality. These included “specialists” and entrepreneurs in Narva small and large enterprises. The youth element was, first of all, students and schoolchildren as future active users of the web-based information systems; active members of the town civic society, including housewives, members of local social organizations, and NGOs.

This study of interests of the stakeholders showed that the main expectation of the new interactive web service is that it will manage to include as much geographical area as possible: from the industrial areas around Narva to private land areas used by Narva inhabitants for gardening and subsistence agriculture and the recreational areas. In the opinion of the interviewed, the information system should focus on presenting aesthetic aspects, such as the bird’s eye view city tour, an opportunity to wander the streets of the old city, and practical aspects – price of land, landowners, contacts of companies renting this or that building, etc. As most Narva residents are Russian-speaking people, the web site should be published in Russian. However, the desire for European integration, and the fact that we live in Estonia, makes it essential that the English and Estonian languages should be added to the web-site design.

Representatives of the stakeholder groups demonstrated great interest in possibilities to use, in the future, the interactive town planning system to be developed. Many of the interviews’ participants stressed that the eCommunity project success can be ensured by involvement of different population groups in consultations during the project implementation and by regular information on the project’s progress in the mass media. Much importance was placed on public events and improving the communication between city authorities and residents, which in the opinion of the interviewed was unsatisfactory.

The main concern of the city residents had to do with the lack of initiative on the part of the local authorities as well as with the lack of a clear urban development plan. The second major problem was considered to be unemployment. The main reason for this was seen in the lack of detailed economic planning and urban development projects. Unemployment is also affecting the attitude of the population, promoting distrust in the future and indifference.

It is clear, that the success of the project will depend on the level of political commitment by the local authorities to the involvement of local stakeholders into the decision-making process

regarding issues of town planning. Only the commitment of the municipality to an open, transparent planning process and readiness to put in time, human and financial resources to the regular communication with the town citizens is a main factor of the project's success.

Gulnara Roll, Peipsi CTC

Startup of the project

The eCommunity project team organized the *first* meeting in Narva, Estonia, on November 21 – 23, 2002. Altogether 39 persons attended the meeting. From eCommunity Consortium, 15 persons attended the meeting, the rest were local stakeholders and members of the Advisory Committee.

The meeting lasted for three days and was chaired by the representative of the beneficiary, Mr. Rauno Schults, from Narva City Government. As it was the first meeting, there were presentations on the deliverables and milestones, also project team (PIU) discussed and detailed working plan?. The PIU also chose the leaders for each task and organized the working groups. All the participants were very pleased with the meeting, which was a good starting point for work to last for three years. The meeting was organized by Hendrikson & KO and hosted by Narva City Government.
The *second* meeting of the project team was called Technology Day and was held in Athens, Greece, on March 02 – 03, 2003. The project partners mostly involved with the technical development of the project attended the meeting, altogether 13 persons.

The key outputs of the meeting were as follows:

- Fixed platform – the fundament on which to create the system
- A common understanding between partners about different sub-tasks
- Setting up of a more precise timetable for sub-tasks
- A common understanding of system and user requirements, technical specifications and technical limitations.

List of partners and contacts

Narva Municipality
Contact person: Rauno Schults
e-mail: rauno.schults@narvaplan.ee
WWW: http://www.narvaplan.ee/

Network Models
Contact person: Varnavas Sergides
e-mail: v.serghides@imperial.ac.uk

Hendrikson & Ko.
Contact person: Vahur Sikaste
e-mail: vahur@hendrikson.ee
WWW: http://www.hendrikson.ee/

Peipsi Center for Transboundary Cooperation
Contact person: Erkki Vedder
e-mail: erkkii@ctc.ee
WWW: http://www.ctc.ee/

Infinity Ltd.
Contact person: Levente Csupor
e-mail: csupor.levente@infinity.co.hu
WWW: http://www.infinity.co.hu/

Sustainable Europe Research Institute
Contact person: Roman Mesicek
e-mail: roman.mesicek@seri.at
WWW: http://www.seri.at/

IDEC
Contact person: Dimitris Kamenopoulos
e-mail: info@idec.gr
WWW: http://www.idec.gr/

Forthcoming events

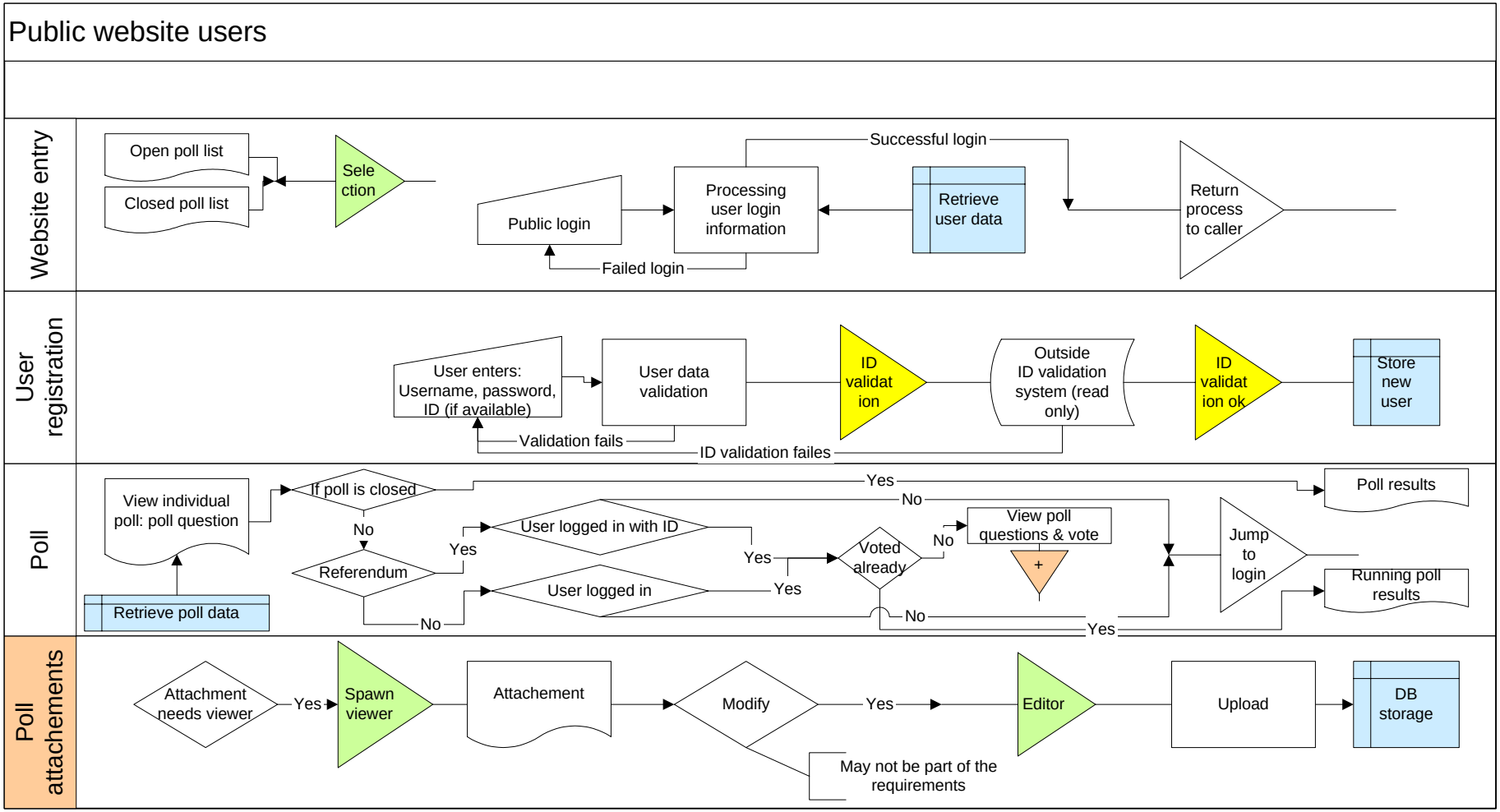
AUGUST 2003	Draft of next modules
SEPTMEBER 2003	Intelligent Meeting – Meeting of project partners and discussion of amendments to the modules. Draft of next modules
FEBRUARY 2004	Meeting with stakeholders in Narva Municipality

Website

The first summary reports and additional information about the project are available on the project website: www.narvaplan.ee/e-com.

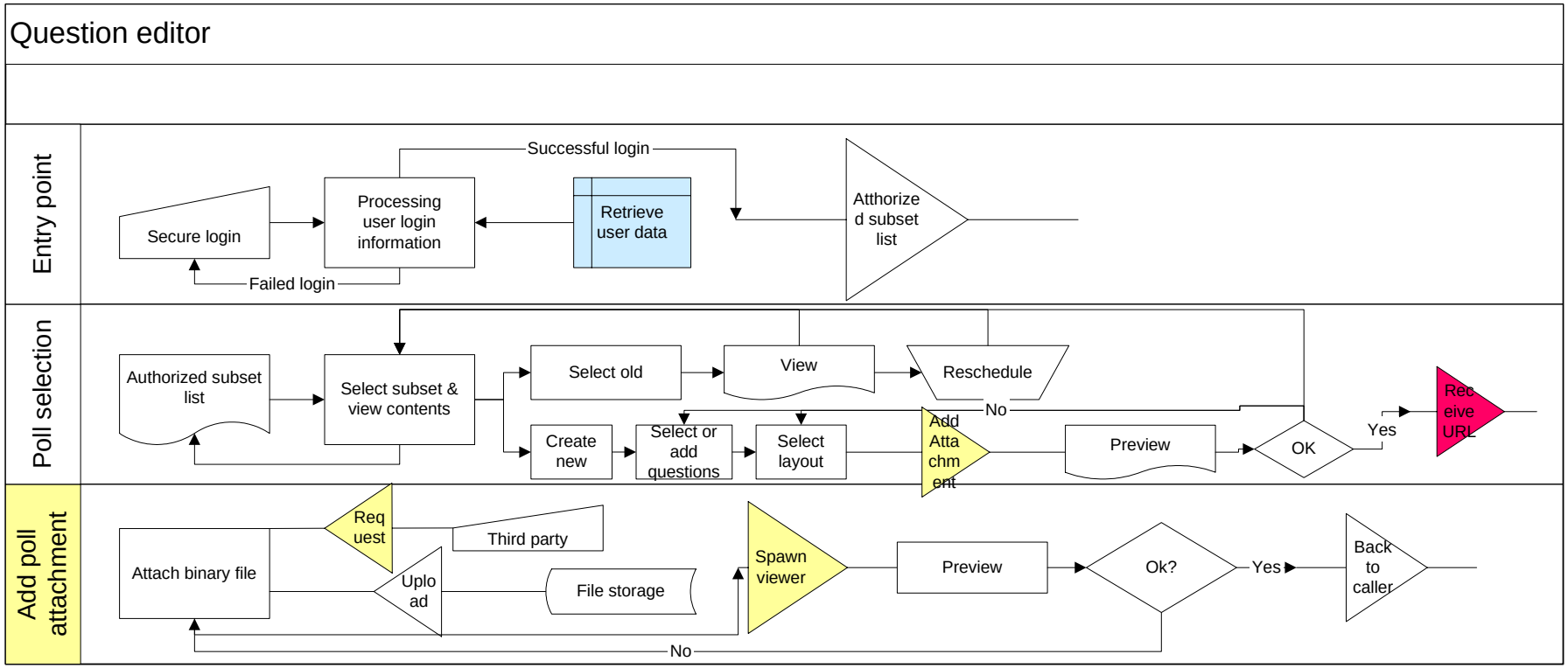
ANNEX 7

System design and data flow model with emphasis on systems integration (v 0.1, February 18, 2003)



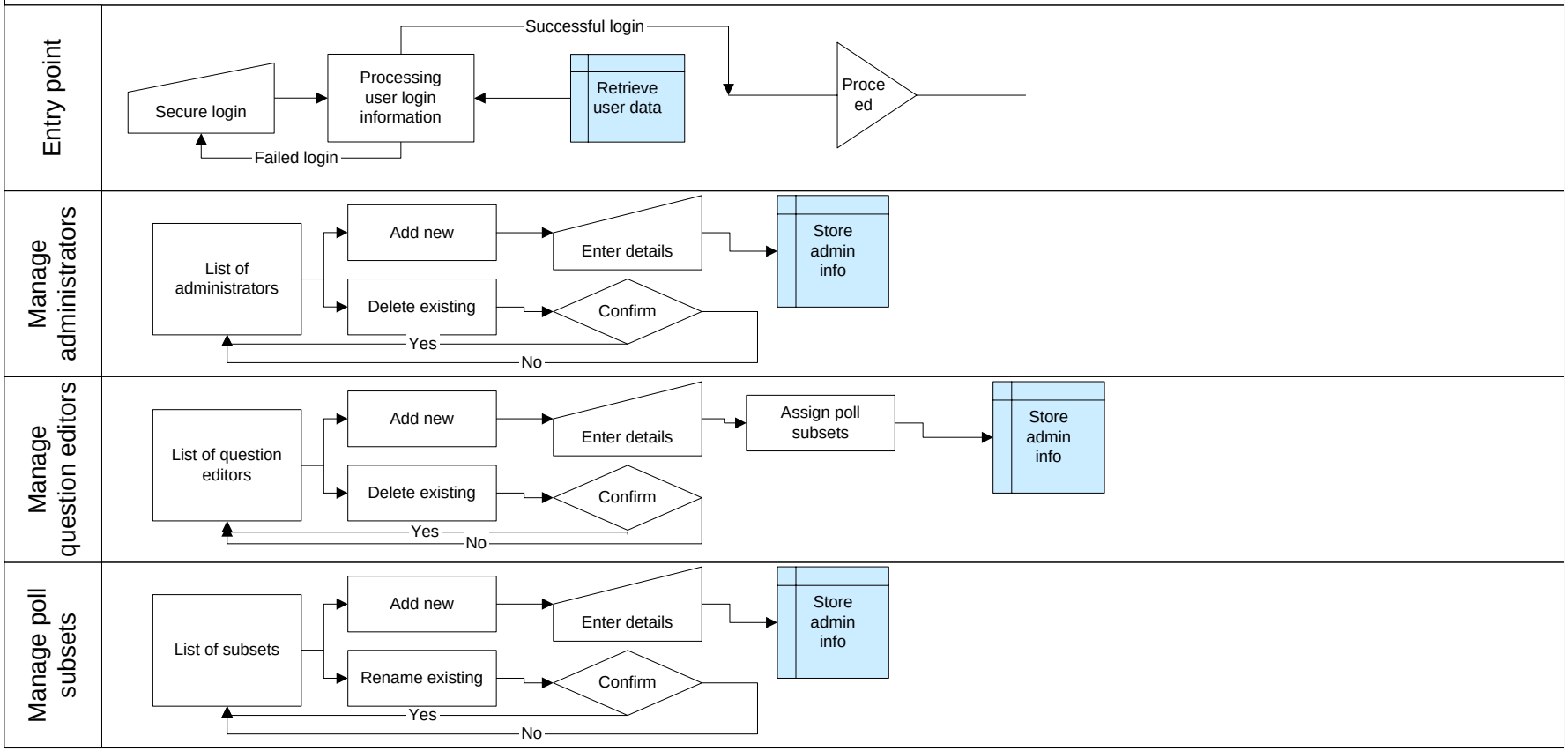
Colored items are attachment point between different systems:
Blue: Database – A lot more attachment point exists, only a few are displayed here
Yellow: ID validation – This is a call to an outside online system

Orange: Poll attachment technology – Includes questions from how the poll attachments are created to how and whether they can modified online
Green: Poll attachment display – Includes questions from poll attachment file format to the technology used to display inside or outside the browser



Colored items are attachment point between different systems:
Blue: Database – A lot more attachment point exists, only a few are displayed here
Yellow: Poll attachment technology – Includes questions from how the poll attachments are created to how and whether they can modified online
Red: Content management – Includes questions on how the retrieved URL will be incorporated into the site, should special layout or display requirements arise

Administrative editor



Colored items are attachment point between different systems:
Blue: Database – A lot more attachment point exists, only a few are displayed here

Other systems integration issues

What are the proposed hosting arrangement (UNIX or Windows based)
What are the proposed coding languages (Web interface)
How do the propsed web interfaces loo like (Usability issue)
What are the proposed database access methods and whether everyone who needs to access database can use a common database?

ANNEX 8

eCommunity Community Day

21 February 2003
Narva Town Hall, Estonia

Participants:

1. Rauno Schults	- Narva City Government
2. Vahur Sikaste	- Hendrikson & Ko
3. Tanel Mazur	- Narva City Government
4. Anne Hallik	- Narva City Government
5. Peep Leppik	- Hendrikson & Ko
6. Tanel Dovnar	- Hendrikson & Ko
7. Tarmo Pikner	- Hendrikson & Ko
8. Uuno Vallner	- Governmental Information Systems Department (RISO)
9. + 18 persons	- Local stakeholders

Main discussion points of the meeting:

1. General vision of the eCommunity project
2. Security of the system and how ID-card systems are connected to eCommunity
3. Discussion

1. General vision of the eCommunity project

Tanel Mazur who is responsible for the project at every day level in Narva City Government presented the structure and main ideas of the project. The project implementation unit (PIU) has described earlier the main ideas and on the seminar the PIU hoped to get some input from local stakeholders. The main proposals:

- If the system is almost ready then to collect again ideas from the public as at the moment the idea was not very understandable to everyone.
- It would be very nice to involve as much as possible from Narva City into those virtual environments.

2. Security of the system and how ID-card systems are connected to eCommunity

Mr. Uuno Vallner from RISO presented the ID-card project to local stakeholders and explained the methodology and possibilities of the connection strategies of the ID-card and eCommunity projects. This was the first time then Governmental ID-card project was introduced to Narva citizens.

3. Although the discussion with the local stakeholders was quite brief one the main output was that local people are very pleased that such project like eCommunity is in Narva. They are also very interested to see the outcomes of the project.
4. The next meeting is taking place in Narva on the 16 of February 2004.