Министерство образования Российской Федерации
Петрозаводский государственный университет
Математический факультет
Кафедра Информатики и математического обеспечения

Сало Андрей Юрьевич

The Web-SynDic project: input data translation, session management, and activity statistics

510200 — Прикладная математика и информатика

Выпускная квалификационная работа бакалавра

Научные руководители:

к.т.н., доцент,
Богоявленская О. Ю.
к.ф.-м.н., ст. преподаватель,
Корзун Д. Ж.

Петрозаводск — 2004

Contents

Introduction

The Web-SynDic project is a student software engineering (SE) project of the Petrozavodsk State University (PetrSU), Department of Computer Science (CSDept). The project is also held in the framework of cooperation between the CS Departments of PetrSU and the University of Helsinki (UH). CSDept of UH helps the PetrSU student team to study SE standards and technology and makes an expert estimation of the process.

The project is related to the research done at CSDept of PetrSU. The research deals with the development of a new type of algorithms for efficient solving some classes of nonnegative linear Diophantine equations (NLDE) by syntactic (parsing) methods.

These syntactic algorithms, developed at CSDept, seem to be promising tool for solving some classes of NLDE system — more exactly a class of NLDE system, associated with formal grammars (ANLDE systems). The algorithms allow efficient (polynomial) computations comparing with the general NLDE case when the same problems are NP-complete or even overNP.

The customer is CSDept of PetrSU, whose representative is the head of CSDept Dr. Yury Bogoyavlenskiy. The key aim is to design and implement a working version of a web system for visual demonstrating and testing the syntactic algorithms via the Internet. The objectives of the project include the following.

- A need of a web system to present current research results on the syntactic algorithms;
- A practical exercise for PetrSU students in software engineering standards and technology;
- Training the PetrSU students to participate in a joint distributed (via Internet) SE project with UH students (it is started at January 2004).

The web system must be developed according with international principles, practice, standards and recommendations of SE. The Project starts on 16.07.2003. Working language of the Project is English. All artifacts are replicated in Russian. From the start of the project, students have to work 20 hours/week.

The waterfall model is chosen for the software process. The customer may slightly change the requirements during "Requirements analysis" and "Design" phases; after that the requirements are frozen. Requirements analysis was performed by the whole team; the results are presented in chapter 1.

During "Design", "Implementation", and "Testing" phases each developer has a number of assigned system modules. My part consists of the following modules:

1. Input data translators (chapter 2);
2. Session management subsystem (chapter 3);
3. Activity statistics subsystem (chapter 4).

The goal is to design, implement, and test the modules.

1 Problem & Software Domain

This section describes problem domain for the software. Several models of the problem domain are presented.

In diagrams,different colors are used to emphasize roles of each problem domain object. The convention about colors, used in models, is stated in Figure 1. The Web-SynDic area defines

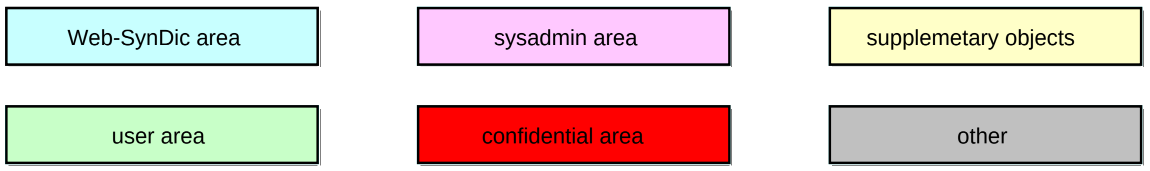


Figure 1: Color convention for models

general bounds of the Web-SynDic system. The user area contains objects that user may have access to. The sysadmin area is for activity of the system administrator. The confidential area must not be accessible by a user (e.g. the external algorithms).

1.1 Structure of the problem domain

The problem domain can be divided into three parts: 1) the user side contains users with a standard Internet browsers, 2) the Web-Syndic system does the required processing, and 3) the external algorithms are called by the Web-SynDic system for generating and solving test ANLDE systems. The described structure is shown in Figure 2.

A user starts a session with Web-SynDic using a standard Internet browser. The web system consists of three subsystems.

1. ANLDE system processing. A user inputs test ANLDE systems for solving and receive the results (see Req. A.1). This is the primary user activity.

2. Supplementary actions. A user may require some additional functions as stated in the User Requirements (see Req. A.1, and A.3). This is the secondary user activity.

3. Users management and administration (see Req. A.1, and A.3). This is area of the system administrator.

Functions of each subsystem are presented in Figure 2 inside the boxes. References to the expanded user requirements are given for each function in brackets.

For ANLDE system processing an execution of the external algorithms are required. They include solvers (for solving test ANLDE systems) and generators (for automatic generation of test ANLDE systems). These algorithms are not a part of Web-SynDic. The goals of Web-SynDic is only to demonstrate and test their work.

User's interactions with Web-SynDic are split into sessions. During a session a user may define (manually or automatically) and solve test ANLDE systems and/or may perform supplementary actions. System administrator may also perform management actions. The actions may be interleaved. A session is implicitly established by a user when she/he has started a Web-SynDic client via the browser; the session is terminated when the client has ended.

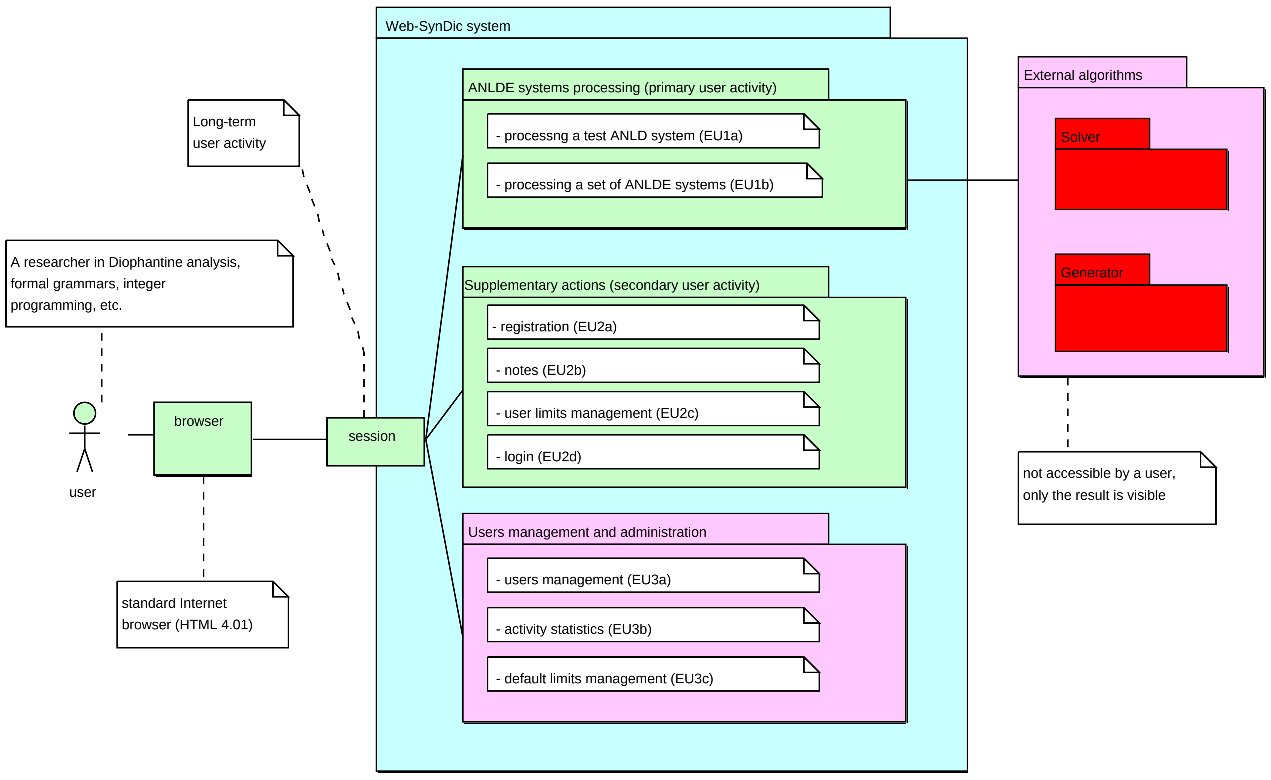

Web-SynDic system
ANLDE systems processing (primary user activity)
- processng a test ANLD system (EU1a)
- processing a set of ANLDE systems (EU1b)
Supplementary actions (secondary user activity)
- registration (EU2a)
- notes (EU2b)
- user limits management (EU2c)
- login (EU2d)
Users management and administration
- users management (EU3a)
- activity statistics (EU3b)
- default limits management (EU3c)
External algorithms
Solver
Generator
not accessible by a user, only the result is visible
Long-term user activity
A researcher in Diophantine analysis, formal grammars, integer programming, etc.
user
browser
session
standard Internet browser (HTML 4.01)

Figure 2: Structure of the problem domain

1.2 Conceptual models and glossary of the problem domain

1.2.1 High-level objects

According with the structure of the problem domain, the following high-level entities and relations can be listed.

Algorithm server (or Web-SynDic algorithm server). Entity. A part of a Web-SynDic server for execution of the external algorithms.

Browser (or Standard Internet browser). Entity. HTML 4.01 must be supported (e.g. Microsoft IExplorer, Mozilla, Netscape, etc.).

Client (or Web-SynDic client, or web client). Entity. A client part of the Web-SynDic system. It is started by a user via browser. Life-time of a client determines user's session.

External algorithm (or demonstrated&tested algorithm). Entity. Solvers (for solving test ANLDE systems) and generators (for automatic generation of test ANLDE systems). These algorithms are not a part of Web-SynDic and are not accessible for users. The goals of Web-SynDic is only to demonstrate and test the work of these algorithms.

Server (or Web-SynDic server). Entity. A server part of the Web-SynDic system. It coordinates and performs the processing. It consists of a web server and algorithm server.

Session (or user session). Relation. It defines long-term user activity with the server. A session is established and terminated implicitly by a user according with the corresponding client.

User (or Web-SynDic user). Entity. A researcher or a person with an interest in Diophantine analysis, formal grammars, integer programming, or related areas.

Web-SynDic (or Web-SynDic system, or web system). Entity. The anticipated software system. The same name is used for the project.

Web server (or Web-SynDic web server). Entity. A part of a Web-SynDic server for supporting all web-related services.

The composition model for these objects is shown in Figure 3. A user starts the client

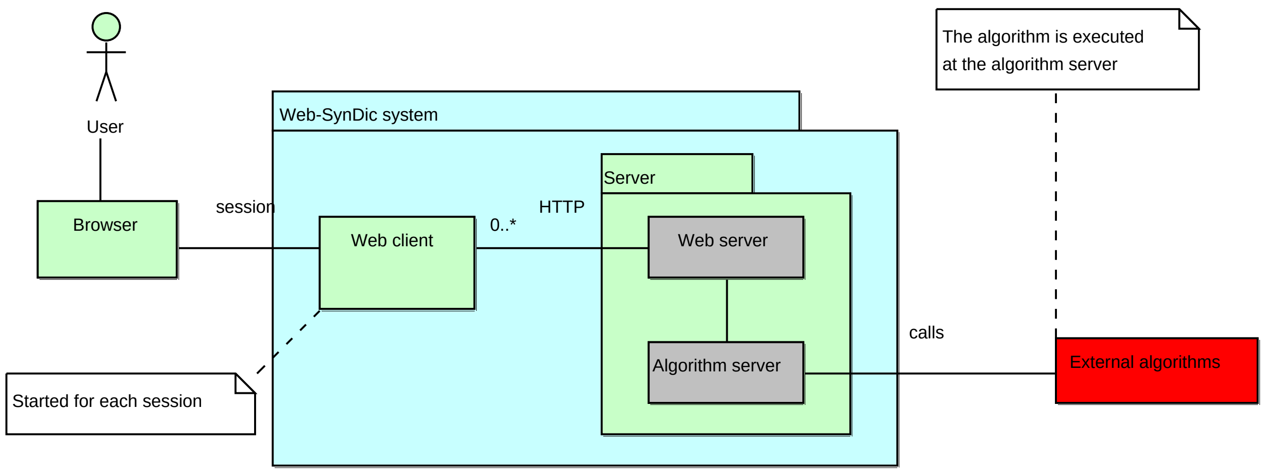


Figure 3: The composite model for high-level objects of the problem domain

via browser. This establishes the session. The client acts as interface to the web server. The web server receives input from the user and communicates, if necessary, with the algorithm server by calling the external algorithms. The web and algorithm servers compose together the Web-SynDic server.

1.2.2 ANLDE system processing

ANLDE systems and related objects form the most complicated part of the problem domain. More details can be found in the Maintenance Document.

ANLDE format Entity. Traditional mathematical style for NLDE system presentation in text files. For instance:

TXT format	formula

`x1 + x2 = 2x1 + 3x3`
`x3 + x4 = x1 + 2x2 + x3`
	$\begin{cases} x_1 + x_2 = 2x_1 + 3x_3 \\ x_3 + x_4 = x_1 + 2x_2 + x_3 \end{cases}$

ANLDE system Entity. NLDE system, associated with a CF-grammar:

$$\mathrm{E}(I)x - Ax = b\ .$$

ANLDE systems set (or set of ANLDE systems). Entity. A set of test ANLDE systems:

$$\big(\mathrm{E}^{(1)}(I^{(1)}) - A^{(1)}\big)\,x = \mathbb{O}\ ,\quad \big(\mathrm{E}^{(2)}(I^{(2)}) - A^{(2)}\big)\,x = \mathbb{O}\ ,\quad \ldots\ ,\quad \big(\mathrm{E}^{(\mathrm{N})}(I^{(N)}) - A^{(N)}\big)\,x = \mathbb{O}\ .$$

Usually the set is generated automatically by a generator and used for detailed analysis of efficiency of a demonstrated/tested solver.

CF-grammar (or grammar). Entity. Formal context-free grammar. It is used for constructing ANLDE systems.

Default limits Entity. Limits on solution process (see Req. AS4 [A.3]). They include maximum time, memory, absolute values of coefficients, maximum number of equations, unknowns, size of ANLDE systems set, solutions in Hilbert basis, maximum values of components of a basis solution.

Hardware config Entity. Configuration of the hardware used for the algorithm server (see Req. F1b.5 [A.1] and F2b.4 [A.1]).

Hilbert basis Entity. The unique finite basis of a test ANLDE system. It consists of all minimal solutions:

$$\big\{h^{(1)}, h^{(2)}, \ldots, h^{(q)}\big\} \subset \mathbb{Z}_+^m.$$

Generator Entity. The program for generating test ANLDE systems and/or their Hilbert bases.

Minimal solution (or undecomposable solution, or basis solution). Entity. A particular solution of a test ANLDE system that cannot be presented as a sum of two nontrivial solutions. All minimal solutions form Hilbert basis.

NLDE Entity. Nonnegative linear Diophantine equation:

$$a_1x_1 + a_2x_2 + \cdots + a_mx_m = b\ ,\quad a_i, b \in \mathbb{Z}\ ,\quad x_i \in \mathbb{Z}_+.$$

Coefficients are integers, solutions are in nonnegative integers.

NLDE system Entity. A system of NLDE:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1m}x_m = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2m}x_m = b_2 \\ \qquad\qquad\cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nm}x_m = b_n \end{cases}$$

or $Ax = b$, where $A \in \mathbb{Z}^{n\times m}$, $b \in \mathbb{Z}^n$, $x \in \mathbb{Z}_+^m$.

Particular solution Entity. Any solution x of a test ANLDE system. The solution can be described as a nonnegative linear combination of basis solutions:

$$x = \alpha_1 h^{(1)} + \alpha_2 h^{(2)} + \cdots + \alpha_1 h^{(q)}, \quad \alpha_s \in \mathbb{Z}_+.$$

Report on solution Entity. The result of a test ANLDE system processing. It may include the test ANLDE system, found Hilbert basis or particular solution, and metrics of solver efficiency (see Req. F1b [A.1] and F2b [A.1]).

Solver Entity. The program for searching Hilbert basis or a particular solution for a given NLDE system.

Solver efficiency Entity. Metrics that show how efficiently a given solver solves a test ANLDE system or ANLDE systems set. This includes time and memory consumption.

Solver outcome Entity. The primary outcome of solver execution. This is used for forming the report on solution.

Test ANLDE system Entity. A homogeneous ANLDE system:

$$\mathrm{E}(I)x - Ax = b\ .$$

It is defined by a user (written manually by a user or generated automatically by a generator). Web-SynDic processes this type of NLDE systems for demonstrating and testing the syntactic algorithms.

User limits Entity. Limits on solution process, defined by a user (see Req. A.3).

The composition model for these objects is shown in Figure 4. A user inputs test ANLDE systems (written manually by the user or generated automatically by a generator). These systems are solved at the algorithm server (Hilbert basis or a particular solution are found). The report on solution includes the found solution (the solver outcome), characteristics of solver efficiency for these ANLDE systems, and hardware configuration of the algorithm server. Solvers are executed according with user limits (or default ones for unregistered users).

1.2.3 Supplementary user actions

IP address Attribute. A particular case of a user ID. It is used for unregistered users.

Nickname Attribute. Unique user ID of a registered user.

Note Entity. An opinion of a user about the Web-SynDic or processing of certain test ANLDE systems (see Req. A.1).

Password Attribute. Password of a registered user (secure).

Registration Relation. A process of user registration (see Req. A.1).

User ID Attribute. Identifier of a regular user. For an unregistered user this is its IP address, for a registered one this is its nickname.

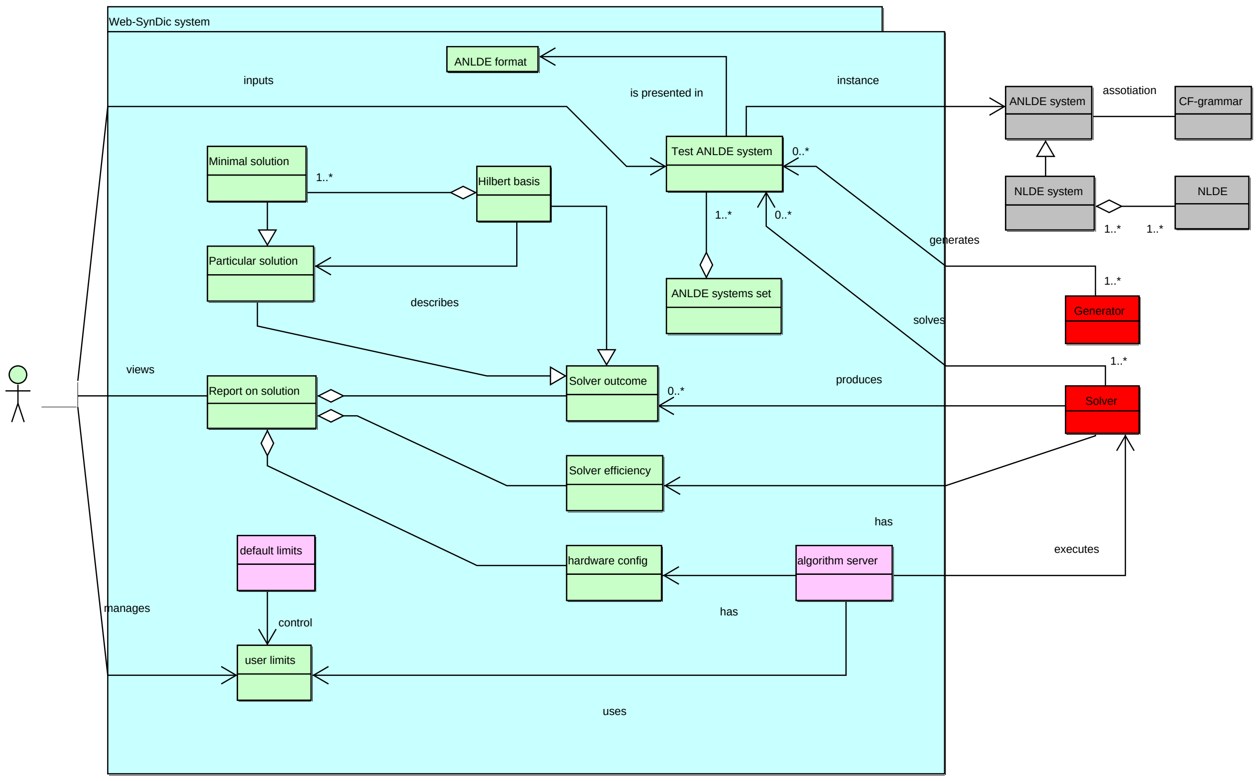


Figure 4: Composition model for ANLDE system processing

User profile Entity. Data for a registered user account (nickname and password).

The composition model for these objects is shown in Figure 5. Any user has user ID (IP address or nickname). A users may define or generated test ANLDE systems for processing. She/He may view the result in reports on solution; solvers are executed at and the reports are formed by the server. She/He may also send notes about Web-SynDic as a whole or about concrete processing. In the latter case, the report on solution can be included to the note. A user may control the solution process by management of her/his user limits.

1.2.4 User management and administration

Activity statistics Entity. Metrics of user activity (see Req. A.1).

Data store Entity. Data on user profiles and user activity.

Registered user Entity. A user who has complete the registration at Web-SynDic and got a user profile (nickname and password).

Regular user Entity. Any user of the Web-SynDic system. She/He is identified by user ID.

Sysadmin (or system administrator, or administrator). Entity. A user who manages and control the Web-SynDic system. She/He maintains the data store, views activity statistics, and manages default limits.

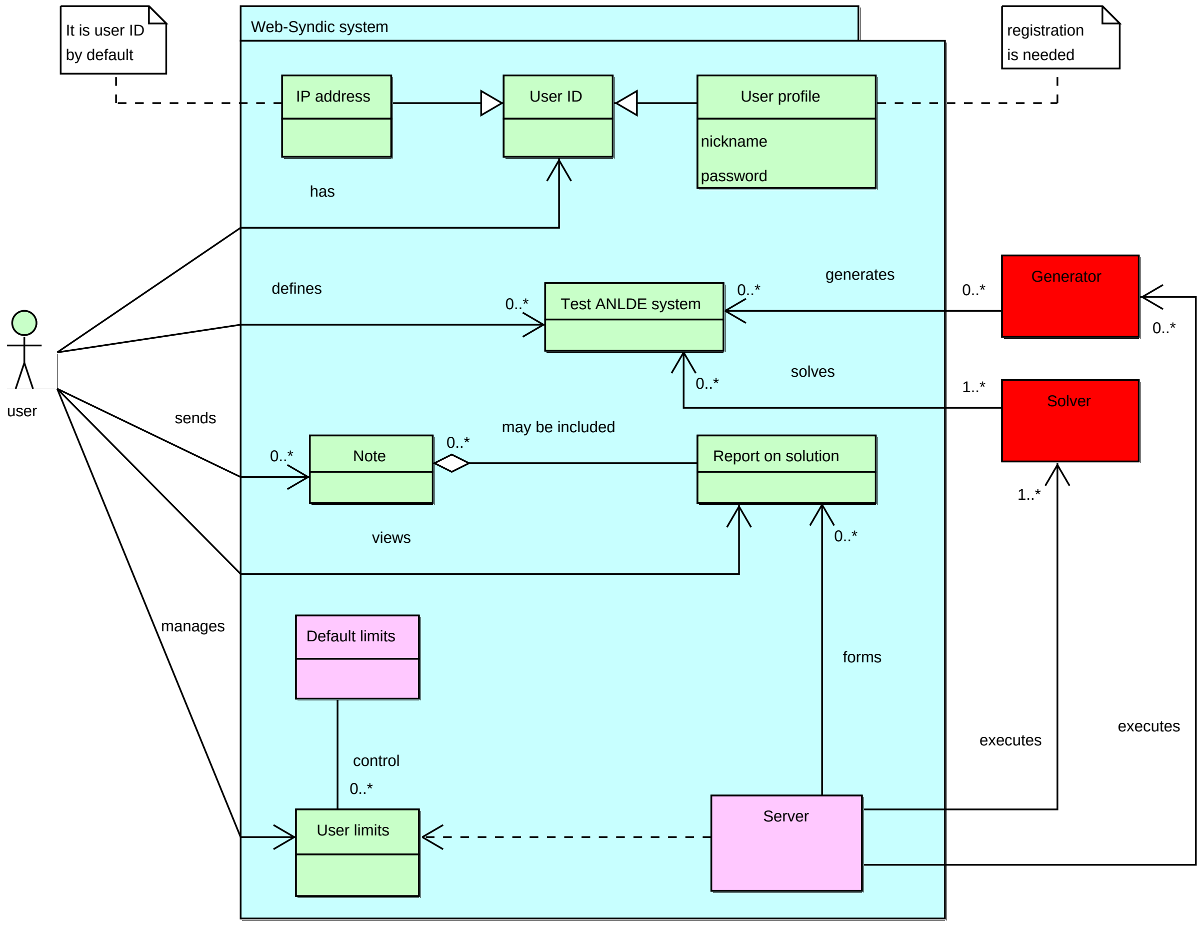


Figure 5: Composition model for supplementary user actions

User activity Entity. Raw history (log) of user actions for producing the activity statistics.

The composition model for these objects is shown in Figure 6. A regular user is identified by IP address and forms user activity. She/He may register and become a registered user with nickname and password. A registered user may manage her/his own user limits. A special user is sysadmin. She/He controls the web system by maintaining the data store (user management) and managing default limits on solution process. She/He may also view activity statistics.

1.3 System Architecture

Initial architecture of Web-SynDic is shown in Figure 7.

The architecture summarizes the essentials of the developed conceptual models of the problem domain (see sect. 1).

Web system can be divided into several high-level modules. At first, the user uses browser to interact with web system. There cannot be any other methods of interaction. Browser uses HTTP application-level protocol to communicate with web system.

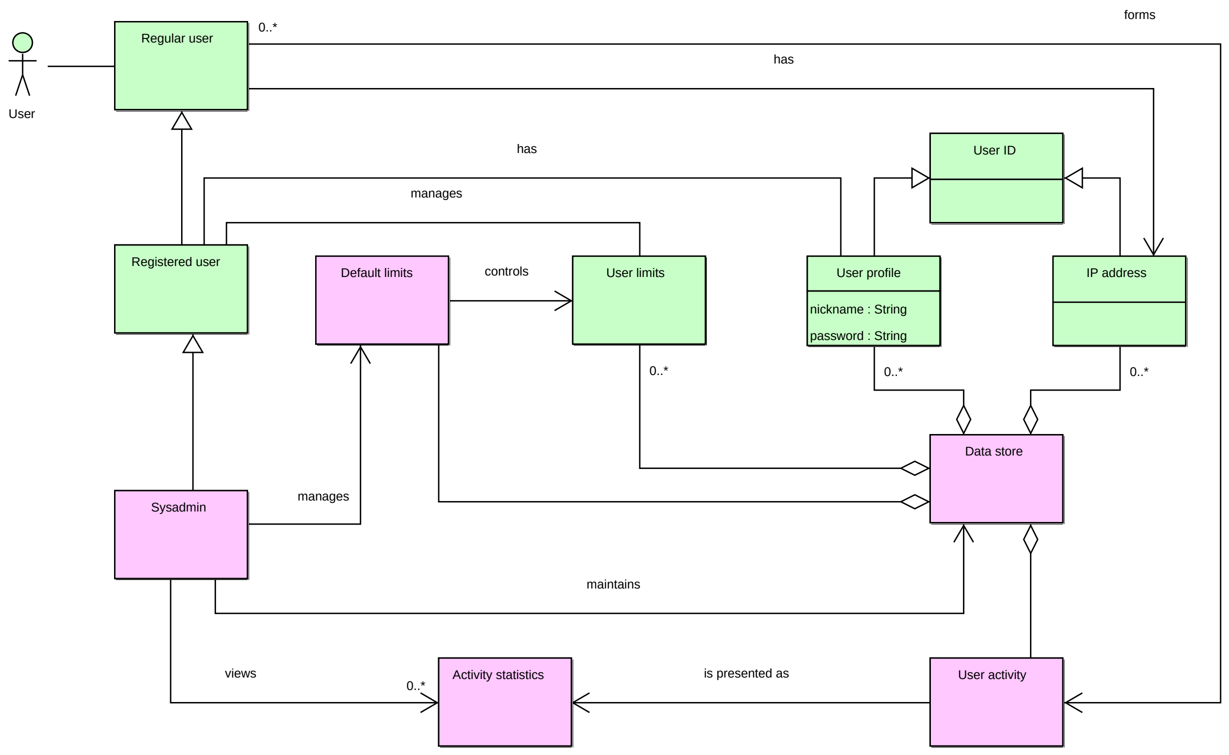


Figure 6: Composition model for user management and administration

It is considered that the client part is a visualization of the problem domain objects (see sect. 1.2.1). It is what the user works with using browser. Actually browser interacts with web server of web system, which is part of the server module destined to maintain all web services. All processing of problem domain objects are also performed by server through algorithm server — the part to execute all external algorithms. Server also performs all management and control functions, and activity statistics evaluation.

All user information (including user limits, activity and profiles of registered users) is stored in the data store, which is managed and kept updated by the server.

Thus, Web-SynDic system is an assembly of client part, server and data store. Browser (and HTTP protocol) are external entities, which interacts with web system. Other external entities are generator and solver of test ANLDE systems, they are not accessible by user. Server communicates with them using algorithm server to execute external algorithms provided by them.

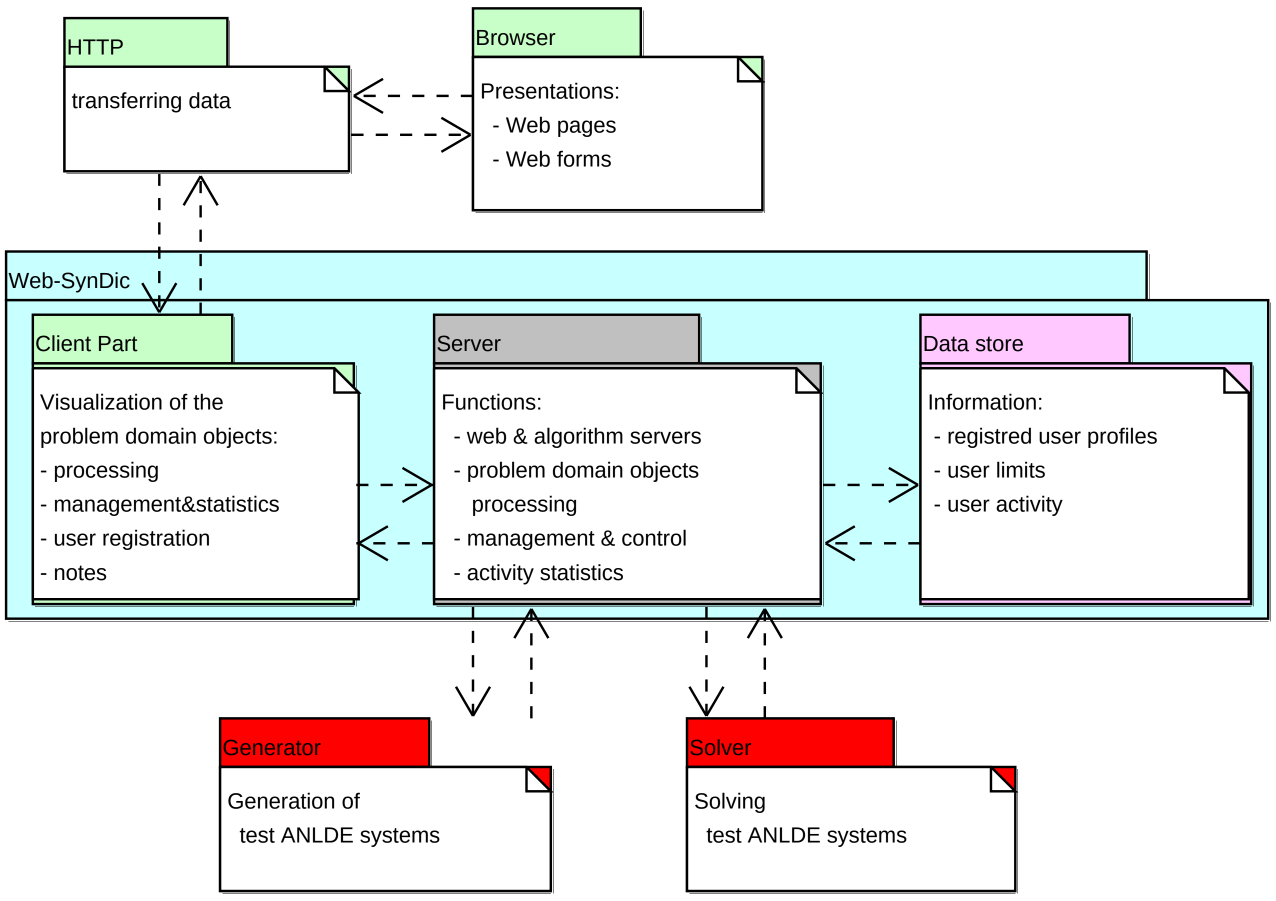


Figure 7: Initial architecture of the Web-SynDic system

2 Input data conversion

One of the most important parts of the user interface is input data conversion. The basic input data consists of ANLDE systems and ANLDE system sets. The input data translators, which are used to convert ANLDE systems and ANLDE system sets from traditional mathematical style to internal memory data structures, are described in this section.

2.1 ANLDE System Format

Format:
```
# Comment
x1 + x2 + ...  + xK2 = c11*x1 + c12*x2 + ...  + c1N*xN
x[K2+1] + x[K2+2] + ...  + xK3 = c21*x1 + c22*x2 + ...  + c2N*xN
...
x[KM+1] + x[KM+2] + ...  + xN = cM1*x1 + cM2*x2 + ...  + cMN*xN
```
Description: The format represents an ANLDE system. c11, c12, ..., c1N, c21, c22, ..., cMN are coefficients (optional, default value is 1). x1, x2, ..., xN are unknowns, may appear in any order, some may be skipped. If there is no unknowns after the "=" sign, write "0". Each unknown must appear in the left-hand side of some equation at most one time. Blank and comment lines are ignored.
Sample:
```
# Sample ANLDE system
x1 + x4 = 2*x1 + 3*x3
x2 + x3 = x1 + 2*x2 + x3
```

2.2 ANLDE System Set Format

Format:
```
<ANLDE system 1>
%
<ANLDE system 2>
%
...
<ANLDE System N>
```
Description: The format represents ANLDE System Set ⟨ANLDE System 1⟩, ..., ⟨ANLDE System N⟩. Each system is in the ANLDE System Input Format (sect. 2.1). Blank and comment lines are ignored. String with symbols(s) '`%`' is a delimiter for ANLDE systems. These strings may additionally contain blank symbols ('␣', '`\t`') only.

2.3 ANLDE System Parser Generation

ANLDE system parser is constructed from two files containing specifications for syntax analyzer and lexical analyzer. Files with specifications are translated to Java source files using Byacc/J and JFlex. The generated class ANLDESystemParser is used by class ANLDEParser to translate single ANLDE systems. The generated class ANLDESystemLexer is used by ANLDESystemParser as a scanner.

Flex specification:

```
INT = [0-9]+                  /* integer value */
VAR = [A-Za-z][A-Za-z0-9]*    /* variable name */
NL  = [\r\n]+                 /* newline */
```

YACC specification:

```
input:                    /* empty input */
    | input line
    ;

line: NL
    | left '=' right NL   /* a line containing an equation */
    | left '=' INT NL     /* a line containing an equation
                             with zero right-hand part */
    ;

left: VAR                 /* rule for left-hand part of ANLDE */
    | left '+' VAR
    ;

right: item               /* rule for right-hand part of ANLDE */
    | right '+' item
    ;

item: VAR                 /* coefficient before variable equals to 1 */
    | INT '*' VAR         /* coefficient before variable equals to INT */
    ;
```

If an error has been matched while parsing, ANLDEFormatException containing error description and line number is thrown. If during the translation coefficients, equations, or unknowns limit has been exceeded, ANLDEFormatException is also thrown.

The generation process consists of two steps:

1. The file containing Flex specification of lexical scanner is passed to JFlex tool and ANLDESystemLexer class is generated.

2. The file containing Yacc specification of syntax analyzer which uses ANLDESystemLexer as lexical scanner are passed to Byacc/J tool and ANLDESystemParser class is generated.

2.4 Classes

The input data conversion subsystem includes a number of classes, which are used to store processed ANLDE systems, and to transfer the data between the other subsystems. The classes, which are used to convert ANLDE systems and ANLDE system sets from traditional mathematical style to internal memory data structures, are generated automatically using JFlex and Byacc/J tools.

Class ANLDE

Description: class for storing a test ANLDE system or a set of ANLDE system.

Fields:

protected ANLDESystem system; — ANLDE system or null if a set of ANLDE systems is stored. protected ANLDESystemSet systemSet; — a set of ANLDE systems or null if ANLDE system is stored.

Constructors:

public ANLDE(ANLDESystem); — constructor for storing ANLDE system.
public ANLDE(); — empty constructor.
public ANLDE(ANLDESystemSet); — constructor for storing a set of ANLDE systems.

Methods:

public abstract void solved(SolverOutcome); — sends solved signal and solver outcome. This method will be implemented in extending classes.
public abstract void generated(); — sends generated signal. This method will be implemented in extending classes.
public abstract void error(String); — sends error signal. This method will be implemented in extending classes.
public void setSystem(ANLDESystem); — sets ANLDE system specified system and ANLDE system set to null.
public void setSystemSet(ANLDESystemSet); — sets a set of ANLDE systems to specifed set and ANLDE system to null.
public ANLDESystem getSystem(); — returns ANLDE system.
public ANLDESystemSet getSystemSet(); — returns ANLDE system set.

Class ANLDESystem

Description: class for storing ANLDE system.

Fields:

private int n; — number of equations.
private int m; — number of unknowns.
private int[n][m] system; — matrix of coefficients.
private int[m] unknowns; — matrix of left part of ANLDE system.

private String[m] names; — list of unknowns names.

Constructors:

ANLDESystem(); — storing ANLDE system

Methods:

public int getN(); — returns number of equations.
public int getM(); — returns number of unknowns
public int[ ][ ] getSystem(); — returns matrix of coefficients.
public int[ ] getUnknowns(); — returns left part of ANLDE system.
public String[ ] getNames(); — returns list of unknowns names.
public void setN(int); — sets number of equations.
public void setM(int); — sets number of unknowns.
public void setSystem(int[ ][ ]); — sets matrix of coefficients.
public void setUnknowns(int[ ]); — sets left part of ANLDE system.
public void setNames(String[ ]); — sets list of unknowns names.

Class ANLDESystemSet

Description: class for storing a set of ANLDE systems.

Fields:

private java.util.List list; — list of ANLDE objects

Constructors:

ANLDESystemSet(); — storing a set of ANLDE systems

Methods:

public int getS(); — returns number of systems.
public java.util.List getList(); — returns set of ANLDE systems.
public void setList(java.util.List); — sets ANLDE systems set.

Class ANLDEParser

Class ANLDEParser is used by web server to translate single ANLDE systems and ANLDE system sets from traditional mathematical style to internal format. Public member ANLDERequest parseANLDE(String source, boolean set, Limits limits) is used to translate ANLDE system (if *set* is false) or ANLDE system set (if *set* is true) to the internal format (instance of ANLDERequest class) from given *source* string according to *limits* specified. ANLDEFormatException is thrown in case of parsing error or when the ANLDE system doesn't conform to the limits.

Class ANLDEFormatException

ANLDEFormatException is thrown by ANLDEParser.parseANLDE() when it fails to parse ANLDE system (set). Public member getMessage() is used to get error description and public method getLine() is used to get line of text where error is found.

3 Session management

3.1 Session concept

One of the main concept of the Web-SynDic system is session. There are two types of sessions: a physical server session and a logical Web-SynDic session. Web-SynDic session stores all information about current work in the system, for example ANLDE systems, user profile etc. Web-SynDic session is a continuous time period of user's working with the Web-SynDic. Further we will use term "session" instead of "Web-SynDic session".

Session is established when any user (regular on registered) start to use the system. It is open while a user logs out or closes the connection. If a user logs out, then the session is closed at once, but if a user closes the connection, for example closes the browser window, the session is still alive for a some period of time (default is 15 min and configured by Jakarta Tomcat server).

If a user has established a session and does not use the system with a large time period the session will be terminated automatically. For using the web-system a user should establish a session again.

Each session has own identifier on the server side. A new identifier of session is generated whenever the session has been established.

If cookies are enabled in user's browser, then the identifier of server session is passed by cookies. If they are disabled the session identifier is passed by the URL string in a browser window.

3.2 Classes

Session processing entity includes one class SessionManager. At the implementation phase, this class will be coded as a part of web server.

Class SessionManager

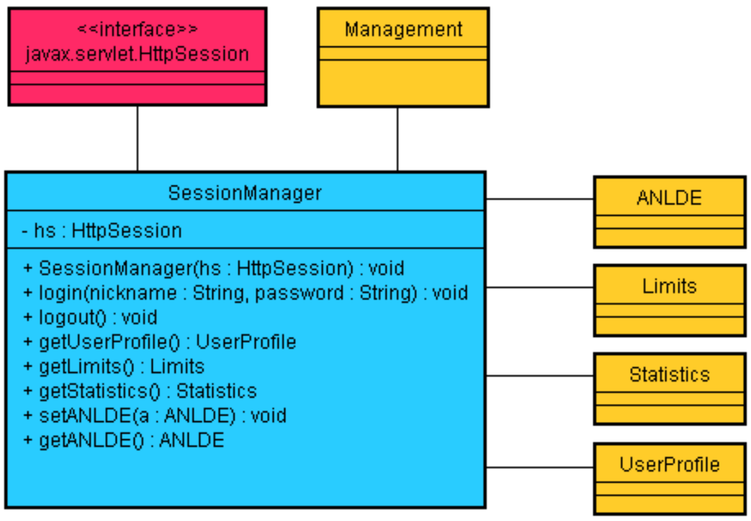
Class SessionManager is used to store user profile, limits, statistics and last solved ANLDE system information. It is shown in Fig. 8.

Fields:

private HttpSession hp; — instance of HttpSession object.

Constructors:

SessionManager(HttpSession hp); — creates a new SessionManager object corresponding to existing http session.



<<interface>>
javax.servlet.HttpSession
Management
SessionManager
- hs : HttpSession
+ SessionManager(hs : HttpSession) : void
+ login(nickname : String, password : String) : void
+ logout() : void
+ getUserProfile() : UserProfile
+ getLimits() : Limits
+ getStatistics() : Statistics
+ setANLDE(a : ANLDE) : void
+ getANLDE() : ANLDE
ANLDE
Limits
Statistics
UserProfile

Figure 8: Class SessionManager

Methods:

public void login(String nickname, String password); — binds attributes of registered user to the existing session:

1. tries to get UserProfile object corresponding to *nickname* and *password*;
2. creates new instance of Statistics object;
3. binds newly created objects to the session. Old objects are unbound automatically and notified using HttpSessionBindingListener;
4. unbinds ANLDE object.

public void logout(); — is similar to login function except that default user's profile is used instead of registered user's profile.
public UserProfile getUserProfile(); — returns user profile.
public Limits getLimits(); — return user limits.
public Statistics getStatistics(); — returns statistics.
public void setANLDE(ANLDE a); — binds new ANLDE object to the session.
public ANLDE getANLDE (); — returns ANLDE object.

4 Activity statistics

Administrator has ability to view activity statistics, collected by the web system. The activity statistics mean summarized information appropriate to selected domain and metrics for current month. Activity statistics domain is a criterion by which statistics information is summarized. Nickname or IP address can be selected as the domain of the report. Activity statistics metrics define categories for statistics report. Number of sessions, total sessions time, requests for solving, requests for generation, total system time, total work time, agreements with solutions can be selected as metrics for a required report. A statistics report contains time of report generation and statistics information appropriate to selected domain and metrics for current month. Records in the report are sorted by domain.

4.1 Activity Domain List Format

Description: The format represents an activity domain list. There are two items in this list: "Nickname" and "IP address". The first one corresponds to statistics by all registered users, the second one is for statistics by all hosts.

4.2 Activity Metrics List Format

Description: The format represents an activity metrics list. The list contains the items:

Sessions is the total number of sessions.

Processed ANLDE systems is the total number of processed (input with an attempt to solve) ANLDE systems,

Generated ANLDE systems is the total number of generated (by Web-SynDic) ANLDE systems,

Solved ANLDE systems is the total number of successfully solved (from point of view of the anlde solver) ANLDE systems,

Acknowledged ANLDE systems is the total number of acknowledged (explicitly by user(s)) systems,

Discrepancies for solvers is the total number of ANLDE systems that are solved with a discrepancy (alternative algorithm gives a different solution comparing with the anlde solver),

Sum system time is sum (for all ANLDE systems and their sets) system (from OS point of view) time usage,

Sum work time is sum (for all ANLDE systems and their sets) work (from user point of view) time usage,

Sum session time is sum session work time (time, used by all sessions),

4.3 Statistics Report Format

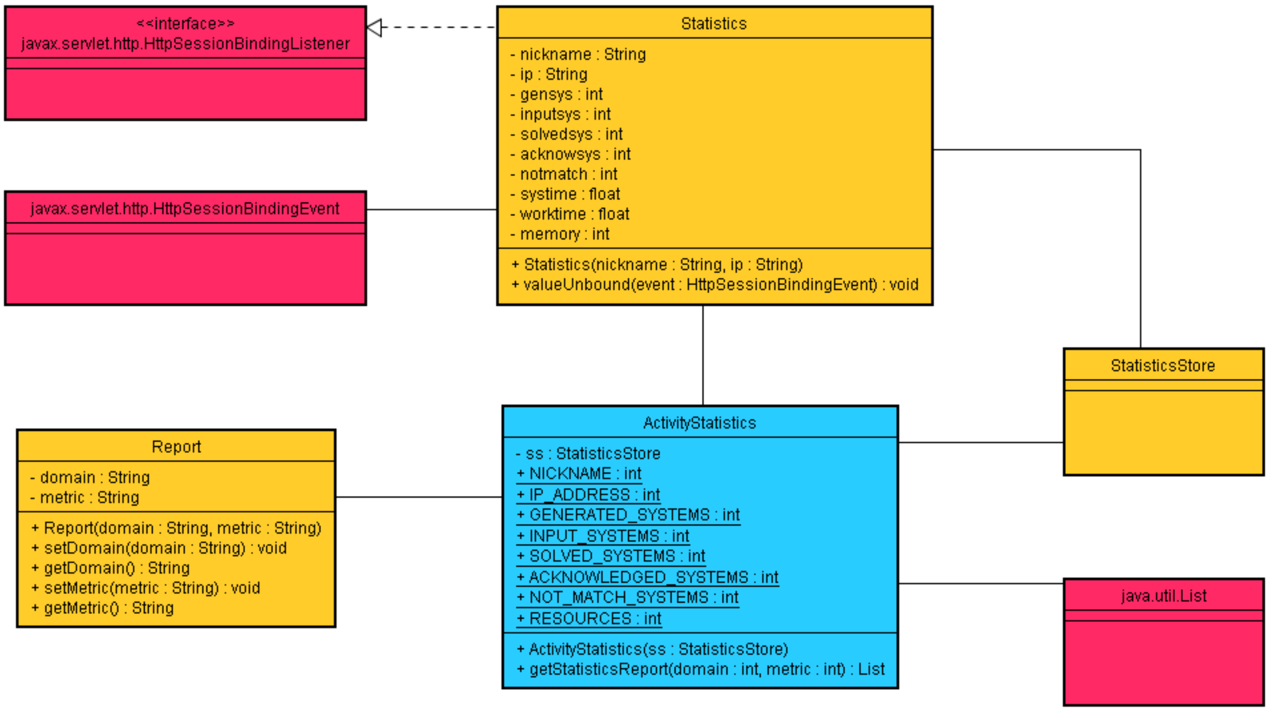
Description: The format represents a statistics report. The report is a table with 2 columns. The first column contains items according to the selected domain (nickname or IP address). The second column contains statistics metrics.
Sample: Statistics report during the October.

Generated at Fri Oct 31 12:22:29 MSK 2003.

nickname	summary used work time
guest	49.40
guest01	10.10
guest2	50.50
guest_2	40.40
user13	20.20
user2	111.00
user21	30.30
user3	50.50
user31	9.0

4.4 Classes

The class diagram used in activity statistics subsystem is shown in Fig. 9.



<<interface>>
javax.servlet.http.HttpSessionBindingListener
Statistics
- nickname : String
- ip : String
- gensys : int
- inputsys : int
- solvedsys : int
- acknowsys : int
- notmatch : int
- systime : float
- worktime : float
- memory : int
+ Statistics(nickname : String, ip : String)
+ valueUnbound(event : HttpSessionBindingEvent) : void
javax.servlet.http.HttpSessionBindingEvent
StatisticsStore
Report
- domain : String
- metric : String
+ Report(domain : String, metric : String)
+ setDomain(domain : String) : void
+ getDomain() : String
+ setMetric(metric : String) : void
+ getMetric() : String
ActivityStatistics
- ss : StatisticsStore
+ NICKNAME : int
+ IP_ADDRESS : int
+ GENERATED_SYSTEMS : int
+ INPUT_SYSTEMS : int
+ SOLVED_SYSTEMS : int
+ ACKNOWLEDGED_SYSTEMS : int
+ NOT_MATCH_SYSTEMS : int
+ RESOURCES : int
+ ActivityStatistics(ss : StatisticsStore)
+ getStatisticsReport(domain : int, metric : int) : List
java.util.List

Figure 9: Activity statistics

Class Statistics

Class Statistics (see Fig. 9) is used to transfer statistics data between ActivityStatistics and StatisticsStore and to store statistics data in SessionManager. Methods set*, add*, and get* are not shown on diagram.

Fields:

private String nickname; — user nickname.
private String ip; — user IP address.

private int gensys; — number of generated systems.
private int inputsys; — number of input systems.
private int solvedsys; — number of solved systems.
private int acknowsys; — number of acknowledged systems.
private int notmatch; — number of ANLDE systems which are discrepances solving.
private float systime; — total system time(sec).
private float worktime; — total work time(sec).
private int memory; — total memory used(Kb).
private long startSession; — start session time mark(ms).
private long endSession; — end session time mark(ms).

Constructors:

Statistics(String nickname, String ip); — constructs new Statistics object with corresponding *nickname* and *ip* and sets other attributes to zero.

Methods:

public void valueUnbound(HttpSessionBindingEvent event); — this function is called when Statistics object is unbound from session. The function extracts StatisticsStore object from servlet context and saves statistics data.

public String getNickname(); — returns nickname.
public String getIp(); — returns ip.

public void setGeneratedSystems(int gensys); — sets number of generated systems.
public void addGeneratedSystems(int gensys); — increases number of generated systems by *gensys*.
public int getGeneratedSystems(); — returns number of generated systems.

public void setInputSystems(int inputsys); — sets number of input systems.
public void addInputSystems(int inputsys); — increases number of input systems by *inputsys*.
public int getInputSystems(); — returns number of input systems.

public void setSolvedSystems(int solvedsys); — sets number of solved systems.
public void addSolvedSystems(int solvedsys); — increases number of solved systems by *solvedsys*.
public int getSolvedSystems(); — returns number of solved systems.

public void setAcknowledgedSystems(int acknowsys); — sets number of acknowledged systems.
public void addAcknowledgedSystems(int acknowsys); — increases number of acknowledged systems by *acknowsys*.

public int getAcknowledgedSystems(); — returns number of acknowledged systems.

public void setNotMatchSystems(int notmatch); — sets number of systems with solutions, which don't match.
public void addNotMatchSystems(int notmatch); — increases number of systems with solutions, which don't match, by *notmatch*.
public int getNotMatchSystems(); — returns number of systems with solutions, which don't match.

public void setSystemTime(float systime); — sets system time.
public void addSystemTime(float systime); — increases system time by *systime*.
public float getSystemTime(); — returns system time.

public void setWorkTime(float worktime); — sets work time.
public void addWorkTime(float worktime); — increases work time by *worktime*.
public float getWorkTime(); — returns work time.

public void setMemory(int memory); — sets memory.
public void addMemory(int memory); — increases memory by *memory*.
public int getMemory(); — returns memory.

public void setStartSession(long startSession); — sets start session time mark.
public long getStartSession(); — returns start session time mark.

public void setEndSession(long endSession); — sets end session time mark.
public long getEndSession(); — returns end session time mark.

Class Report

Class Report (see Fig. 9) is used to transfer processed statistics data (report data) from ActivityStatistics to statisticsreport.jsp.

Fields:

private String domain; — requested domain.
private String metric; — requested metric.

Constructors:

Report(String domain, String metric); — constructs new Report object with corresponding *domain* and *metric* attributes.

Methods:

public void setDomain(String domain); — sets domain.

public String getDomain(); — returns domain.
public void setMetric(String domain); — sets metric.
public String getMetric(); — returns metric.

Class ActivityStatistics

Class ActivityStatistics (see Fig. 9) is used to process statistics data and prepare report according to requested domain and metric.

Fields:

StatisticsStore ss; — instance of StatisticsStore object.
public static final int NICKNAME; — specifies nickname as report domain.
public static final int IP_ADDRESS; — specifies IP address as report domain.
public static final int GENERATED_SYSTEMS; — specifies number of generated systems as report metric.
public static final int INPUT_SYSTEMS; — specifies number of input systems as report metric.
public static final int SOLVED_SYSTEMS; — specifies number of solved systems as report metric.
public static final int ACKNOWLEDGED_SYSTEMS; — specifies number of acknowledged systems as report metric.
public static final int DISCREPANCIES; — specifies number of ANLDE systems, which discrepances solving as report metric.
public static final int USED_SYSTEM_TIME; — specifies summary used system time as report metric.
public static final int USED_WORK_TIME; — specifies summary used work time as report metric.
public static final int SESSION_WORK_TIME; — specifies summary session work time as report metric.
public static final int SESSIONS; — specifies number of sessions as report metric.

Constructors:

ActivityStatistics(StatisticsStore ss); — constructs new ActivityStatistics object.

Methods:

public List getStatisticsReport(int domain, int metric); — prepares statistics report according to requested *domain* and *metric*:

1. tries to get list of Statistics objects from StatisticsStore;
2. processes list of Statistics objects using algorithm:
   (a) get domain value (nickname or IP address) Statistics object (according to *domain* specified);
   (b) find Statistics object with the same domain value;

    (c) add some of found Statistics object metrics values (according to *metric* specified) to initial Statistics object metrics values;
    (d) remove found Statistics object from the list;
    (e) repeat for Statistics objects with other domain values;
3. prepares sorted list of Report objects;
4. returns list of Report objects.

Conclusion

The main results of my participation in the Web-SynDic project are:

1. Input data conversion, session management, and activity statistics subsystems are designed, implemented, and tested according with international principles, practice, standards, and recommendations of SE.

2. Working version of the product is ready. Publication of the Web-SynDic project will be finished in September 2004.

3. The experience in software engineering standards and technology is gained.

The Web-SynDic project was awarded the first place at the conference of Microsoft technologies in theory and practice ([11]). Separate parts of the project were presented at the 56 scientific student conference of Petrozavodsk State University ([12], [13]) and won first and second places.

The Web-SynDic was the training project before the joint distributed (via Internet) SE project with University of Helsinki students. This distributed project named DaCoPAn was started at January 2004 and will be finished in May 2004.

A Collection of the User Requirements

The collection is based on the User Requirements v. 1.20 of the Maintenance Document.

A.1 Functions of the web system

F1a: Processing a test ANLDE system

The web system must solve a test ANLDE system and show to a user the report on solution. A test ANLDE system is given manually by a user or generated automatically by a generator (user choice). Only homogeneous ANLDE are used as test ones for this project.

F1b: Format of a report on solution (test ANLDE system)

For the case of Req. F1a a report on solution must include:

1. Test ANLDE system.

2. Solutions of the ANLDE system (Hilbert basis or a particular solution).
3. Metrics of the resource consumption by the syntactic algorithm (time and memory usage estimates). Concrete metrics will be defined at the design phase.
4. Comparative metrics for the alternative algorithms (slopes, lp_solver). The metrics are the same as for the syntactic algorithms.
5. Key hardware characteristics of the algorithm server.

F2a: Processing an ANLDE systems set

The web system must solve a set of test ANLDE systems (ANLDE systems set). An ANLDE systems set is given by a user (TXT file) or generated automatically by the web system using a generator (user choice).

F2b: Format of a report on solution (ANLDE systems set)

For the case of Req. F2a a report on solution must include:

1. Characteristics of the input set of ANLDE systems. Concrete metrics will be defined at the design phase.
2. Metrics of the resource consumption by the syntactic algorithm (time and memory usage estimates). Concrete metrics will be defined at the design phase.
3. Comparative metrics for the alternative algorithms (see references for these algorithms in Req. F1b). The metrics are the same as for the syntactic algorithms.
4. Key hardware characteristics of the server. The format is the same as for Req. F1b.

F3: User notes

The web system must allow a user to send her/his opinion on the solution result (as a note). A special case is user's explicit agreement/disagreement with the found solution of the processed test ANLDE system (testing of the syntactic algorithms). The test ANLDE system may be included to the opinion (user choice). Notes are sent to the system administrator. Noting is a type of user activity; it must be used for activity statistics (see Req. F5 and F6). Simple TXT format is used for notes but Req. AU1 must be satisfied.

F4: User registration

The web system must register a user when she/he wishes. A registered user has a unique identifier (nick name) and a password. A registered user may log in to the web system and may use additional features (see Req. AS5).

F5: Activity statistics of registered users

The web system must compute activity statistics of registered users. The activity includes: 1) login, 2) requests for solving, 3) noting, 4) successful/unsuccessful solving, 5) resource usage, and 6) ANLDE generating. This function is available for system administrator only.

F6: Activity statistics of regular users

The web system must compute activity statistics of all users (regular users). These users are identified by their IP-addresses. The activity includes: 1) visits, 2) requests for solving, 3) noting, 4) successful/unsuccessful solving, 5) resource usage, and 6) ANLDE generating. This function is available for system administrator only.

A.2 Usability

AU1: Traditional mathematical style

The traditional mathematical style must be supported for representation of ANLDE (and possibly NLDE) systems and their solutions for a user.

AU2: Format of output for a user

Any output of the web system, available to a user, must be in simple HTML and TXT formats.

AU3: Standard Internet browser

Standard Internet browsers (among them Netscape, Mozilla, MS-IExplorer) must be supported. This means HTML 4.01 support.

A.3 Security

AS1: Access to the external algorithms

All external algorithms (including demonstrated&tested syntactic solvers and generators) must not be accessible to the external side. Only the outcomes of their work are available to a user.

AS2: Regular users and sysadmin

There are two types of users: regular ones and system administrator.

AS3: Access to activity statistics

The activity statistics must not be accessible for regular users. See also Req. F5 and F6.

AS4: Default limits

For any regular user the web system must support default limits on the solution process (maximum time, memory, absolute values of coefficients, maximum number of equations, unknowns, size of ANLDE systems set, solutions in Hilbert basis, maximum values of components of a basis solution).

AS5: User limits

A regular user may manage her/his own limits on the solution process; these limits must not exceed the default ones (see Req. AS4).

A.4 Performance

AP1: Concurrent user sessions

The web system must serve concurrently up to 5 users (separate user sessions) without significant reduction of the server performance.

AP2: Web server overload

The web system must not overload a base server more than 75% of the total server workload.

AP3: Notification on the process

The web system must reply on a user action less than after 20 seconds. The reply is either the required data, or a notification on the progress.

A.5 Deployment

AD1: No installation for a client

Client part of the web system must be available to a user via an Internet browser without an explicit installation.

References

[1] Miguel Filgueiras M., Ana-Paula Tomás. *Solving Linear Constraints on Finite Domains through Parsing.* In P. Barahona, L. Moniz Pereira, A. Porto (eds.), Proceedings of the 5th Portuguese Conference on Artificial Intelligence, Springer-Verlag, 1991. LNAI 541, pp. 1–16.

[2] Dmitry G. Korzun. *Syntactic Algorithms for Solving Nonnegative Linear Diophantine Equations and their Application for Modelling of Internet Link Workload Structure.* PhD Thesis, Department of Computer Science, University of Petrozavodsk, 2002. 185 p.

[3] Dmitry G. Korzun. *Grammar-Based Algorithms for Solving Certain Classes of Nonnegative Linear Diophantine Systems.* Proceedings of Annual international Finnish Data Processing Week at the University of Petrozavodsk (FDPW'2000): Advances in Methods of Modern Information Technology. Vol. 3. Petrozavodsk, 2001, pp. 52–67.

[4] Schrijver A. *Theory of linear and integer programming.* Wiley, Chichester, 1986.

[5] Huet G. *An algorithm to generate the basis of solutions to homogenous linear diophantine equations* // Information Processing Letters, 1978. Vol. 3. No. 7. PP. 144–147.

[6] Domenjoud E. *Solving Systems of Linear Diophantine Equations: An Algebraic Approach.* In U. Tarlecki (ed.), Proceedings of 16th International Simposium on Mathematical Foundations of Computer Science. Springer–Verlag, 1991. LNCS 520. PP. 141–150.

[7] Roger S. Pressman. *Software Engineering. A Practitioner's Approach.* European adapt., 5th ed. McGraw-Hill, 2000. 915 p. ISBN 0-07-709677-0.

[8] Ian Sommerville. *Software Engineering.* 6th ed. Addison-Wesley, 2000. ISBN 0-201-39815-X.

[9] Graig Larman. *Applying UML and Patterns.* Prentice Hall, 2000. ISBN 0-13-748880-7.

[10] Jim Conallen. *Building Web Applications with UML.* Addison-Wesley, 2000. ISBN 0-201-61577-0.

[11] Кулаков К. А., Сало А. Ю., Ананьин А. В., Крышень М. А. *Web-SynDic: система демонстрации и тестирования синтаксических алгоритмов решения неотрицательных линейных диофантовых уравнений.* Технологии Microsoft в теории и практике программирования. Материалы межвузовского конкурса-конференции студентов и молодых ученых Северо-запада. СПб.: Изд-во СПбГТУ, 2004. С. 43.

[12] Крышень М. А. *Система Web-SynDic: раработка сервера и интерфейса пользователя.* Материалы 56-й научной студенческой конференции ПетргУ. Петрозаводск: Изд-во Петрозаводского гос. ун-та, 2004. (принято в печать)

[13] Ананьин А. В. *Система Web-SynDic: разрботка программного обеспечения на основе прецедентов.* Материалы 56-й научной студенческой конференции ПетргУ. Петрозаводск: Изд-во Петрозаводского гос. ун-та, 2004. (принято в печать)