Web-SynDic

Web System for Demonstrating the Syntactic Algorithms for Solving Linear Equations in Nonnegative Integers (Nonnegative Linear Diophantine Equations)

Requirements Specification

Department of Computer Science, Petrozavodsk State University, Russia

15th November 2004

Contents

1 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).

2 General Description

This section provides an overview of the entire requirement document.

The document is based on the User Requirements; they were explicitly given by the customer. The key ones are listed in Section 3. The full list is available in the Maintenance Document.

In general, the required functionality can be described as follows. A user gives her/his ANLDE system to the web system; it responds with the solution and some characteristics of the computation. This allows to present key features of the syntactic algorithms, test them, estimate the efficiency, and compare with available alternatives of other authors.

A user is assumed to be a researcher in Diophantine analysis, formal grammars, integer programming, and related fields (or just a person with an interest in the area). She/He has an access to the Internet via a standard browser. This is enough for using the web system; no

special knowledge in software engineering or networking is required. The web system does not allow a user to have a direct access to the algorithms; it only shows an outcome of their work.

The requirement analysis for this project is based on Unified Modelling Language (UML). The ground for the analysis is the collection of the user requirements; the key ones are listed in Section 3 and referenced all through the text.

The detailed analysis of the problem domain is given in Section 4, where several models of the problem domain is developed. This gives a comprehensive view on the solving problem.

The initial system architecture, based on the problem domain model, is presented in Section 5. This gives a high-level overview of the distribution of functions across system modules.

The expanded user requirements are developed in Section 6. They describe the key functionality of the web system and introduce a high-level usage scenario for the web system.

More detailed view on the functions of the anticipated web system is introduced in Section 7. Analysis of the expanded user requirements is resulted in a lot of basic functions/operations. These are the system requirements that implement the required usage scenario.

The most comprehensive view on the functionality (user centric) is presented in Section 8. UML use case models are used to develop this view. This is the most important blueprint of the web system; it should be used as a base model for further development.

Section 9 states the criteria for validating the specified requirements. This can be considered as the primary criteria for recognizing the successful implementation and for accepting the web system.

The appendix includes a specification of the minimal requirements for system configuration (Section A) and a description of essentials of the external algorithms (ANLDE solvers and generators) to be used (Section B).

3 Collection of the User Requirements

The collection is based on the User Requirements v. 1.20 of the Maintenance Document. The list, presented here, must be considered as final and frozen after accepting the Requirement Specification by the Customer and by the project.

3.1 Functions of the web system

3.1.1 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.

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

For the case of Req. F1a [3.1.1] 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, BonsaiG (rejected by the project, see sect. B), and GLPK). The metrics are the same as for the syntactic algorithms.
5. Key hardware characteristics of the algorithm server. See Section A for the format.

3.1.3 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).

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

For the case of Req. F2a [3.1.3] 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 [3.1.2]). 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 [3.1.2].

3.1.5 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 [3.1.7] and F6 [3.1.8]). Simple TXT format is used for notes but Req. AU1 [3.2.1] must be satisfied.

3.1.6 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 [3.3.5]).

3.1.7 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.

3.1.8 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.

3.2 Usability

3.2.1 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.

3.2.2 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.

3.2.3 AU3: Standard Internet browser

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

3.3 Security

3.3.1 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.

3.3.2 AS2: Regular users and sysadmin

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

3.3.3 AS3: Access to activity statistics

The activity statistics must not be accessible for regular users. See also Req. F5 [3.1.7] and F6 [3.1.8].

3.3.4 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).

3.3.5 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 [3.3.4]).

3.4 Performance

3.4.1 AP1: Concurrent user sessions

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

3.4.2 AP2: Web server overload

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

3.4.3 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.

3.5 Deployment

3.5.1 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.

4 Problem Domain Model

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

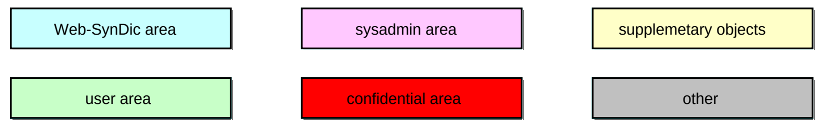

Figure 1: Color convention for models

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).

4.1 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. 3.1.1 and 3.1.3). This is the primary user activity.

2. Supplementary actions. A user may require some additional functions as stated in the User Requirements (see Req. 3.1.5, 3.1.6, and 3.3.5). This is the secondary user activity.

3. Users management and administration (see Req. 3.1.7, 3.1.8, 3.3.2, and 3.3.4). This is area of the system administrator.

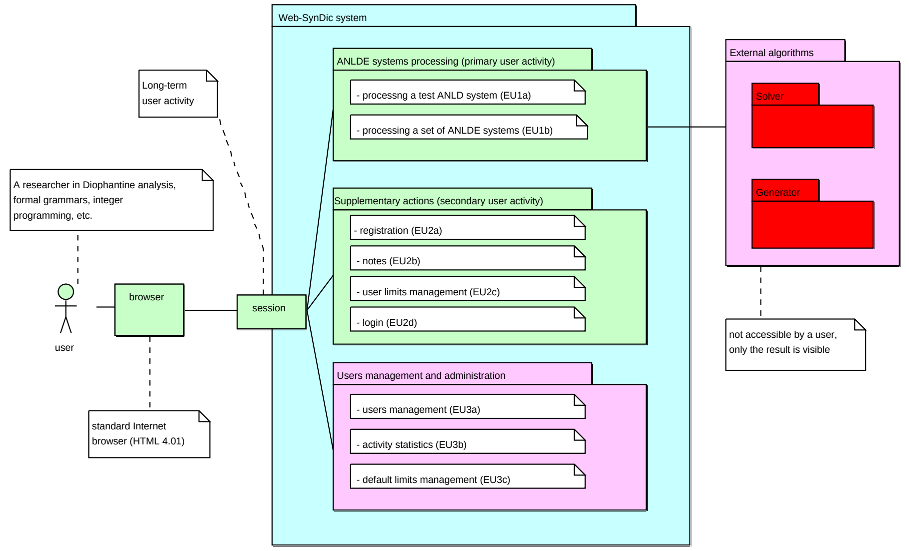
Web-SynDic system
ANLDE systems processing (primary user activity)
- processng a test ANLD system (EU1a)
- processing a set of ANLDE systems (EU1b)
Supplementary actions (secondary user activity)
- registration (EU2a)
- notes (EU2b)
- user limits management (EU2c)
- login (EU2d)
Users management and 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

Functions of each subsystem are presented in Figure 2 inside the boxes. References to the expanded user requirements (see sect. 6) 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.

4.2 Conceptual models and glossary of the problem domain

4.2.1 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 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.

4.2.2 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, section Problem Domain.

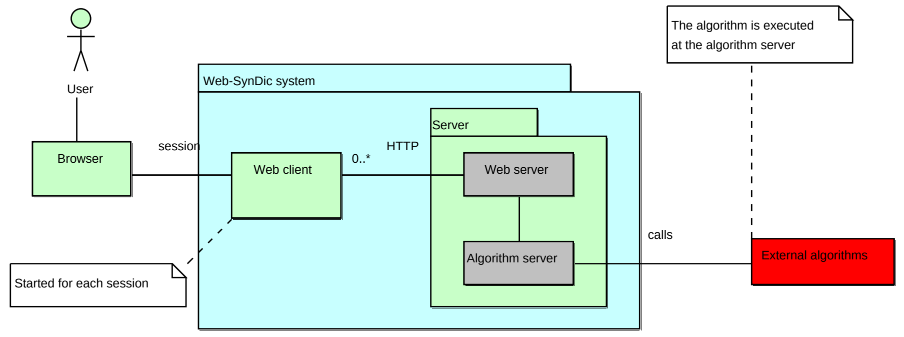

Figure 3: The composite model for high-level objects of the problem domain

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:

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

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 [3.3.4]). 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 [3.1.2] and F2b.4 [3.1.4]).

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

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

Generator Entity. The program for generating test ANLDE systems and/or their Hilbert bases. See section B.3 in the appendix.

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 [3.1.2] and F2b [3.1.4]).

Solver Entity. The program for searching Hilbert basis or a particular solution for a given NLDE system. See section B in the appendix.

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. 3.3.5).

The composition model for these objects is shown in Figure 4. A user inputs test ANLDE

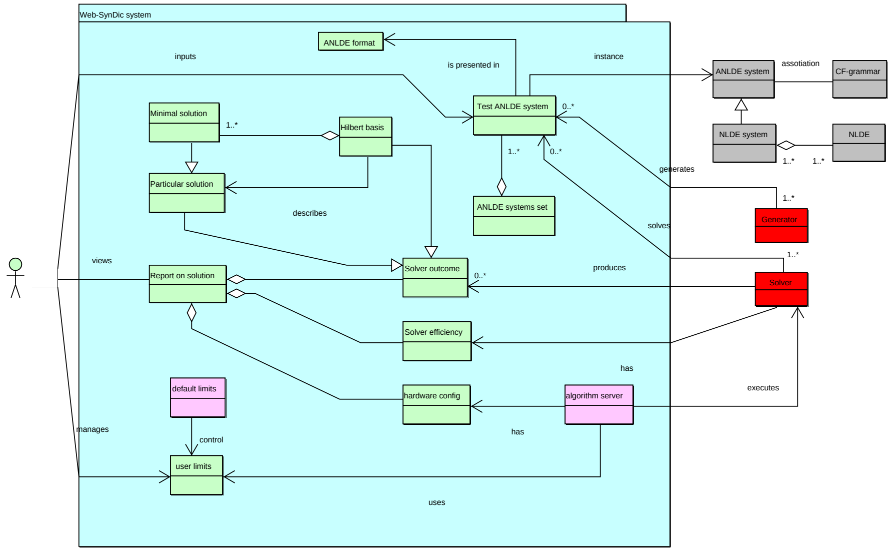

Figure 4: Composition model for ANLDE system processing

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).

4.2.3 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. 3.1.5).

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

Registration Relation. A process of user registration (see Req. 3.1.6).

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.

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.

4.2.4 User management and administration

Activity statistics Entity. Metrics of user activity (see Req. 3.1.8).

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.

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

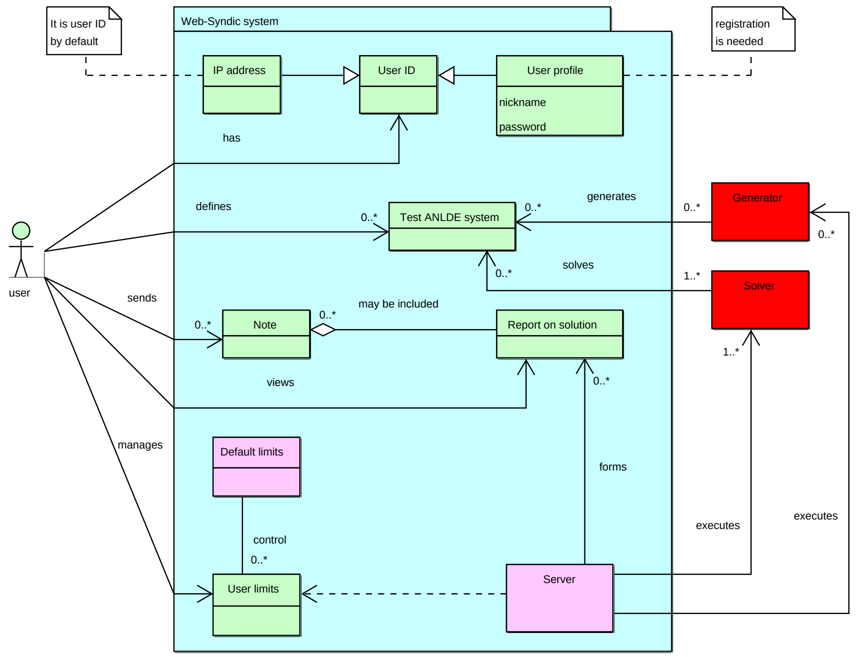

Figure 5: Composition model for supplementary user actions

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.

5 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. 4).

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

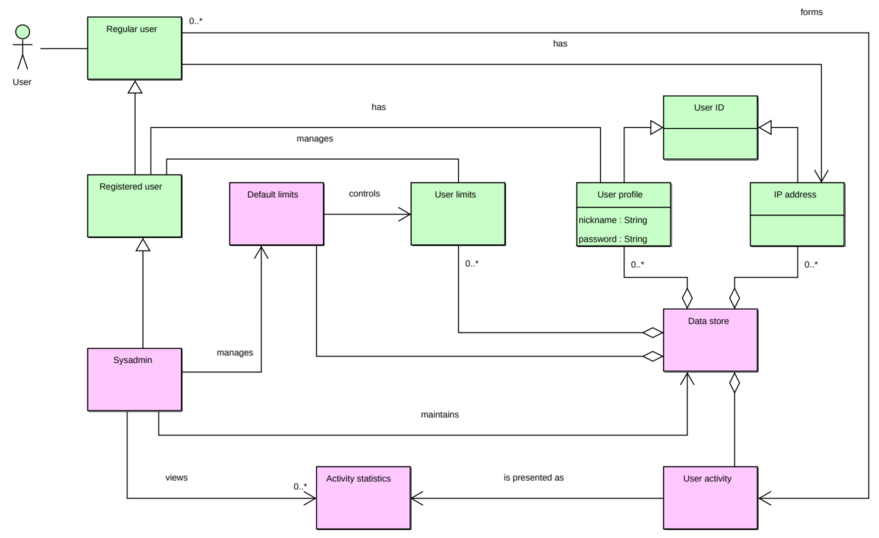

Figure 6: Composition model for user management and administration

HTTP application-level protocol to communicate with web system.

It is considered that the client part is a visualization of the problem domain objects (see sect. 4.2.1). 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 processings 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.

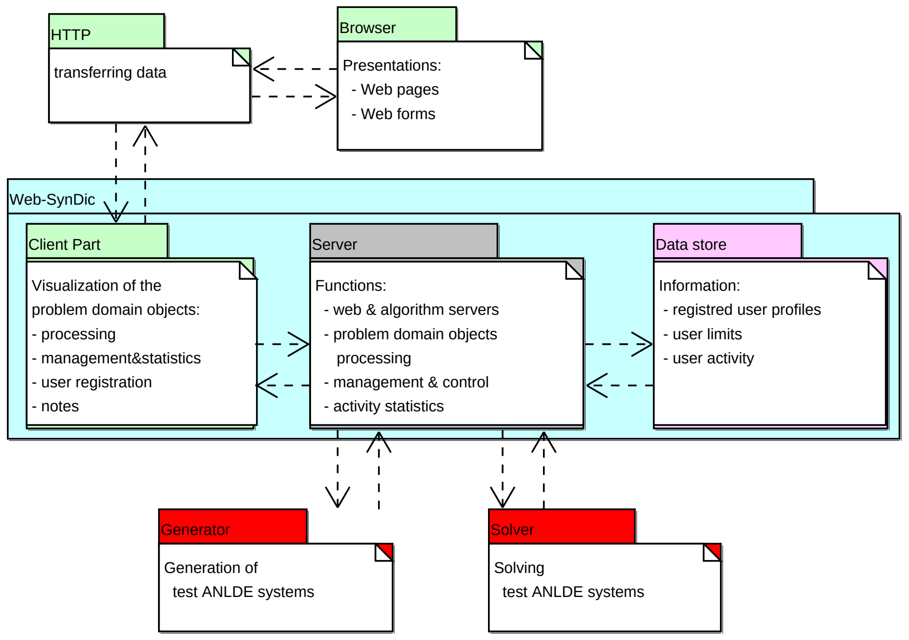

Figure 7: Initial architecture of the Web-SynDic system

6 Expanded User Requirements

The expanded user requirements describe the required functionality of the web system.

6.1 EU0: User session starting and finishing

Description

A user starts a session, open form in the required processing, and then he/she finishes the session.

User Actions

1. Start a client (AU3 [3.2.3], AP1 [3.4.1], AP2 [3.4.2], AD1 [3.5.1]).
2. Use the web system.
3. Close a client (this a resporcifility of a browser; the user just close the web page).

Rationale

All user activity is splitted into sessions. A user must perform the required processing only during a session with Web-SynDic.

References

System requirements: startSession, sect. 7.1; finishSession, sect. 7.2.
Use cases: work with Web-SynDic, sect 8.1.

6.2 EU1a: Solving a test ANLDE system

Description

The web system solves a test ANLDE system and produces the report on solution.

User Actions

1. Start the ANLDE processing subsystem, select solver (AU3 [3.2.3], AP1 [3.4.1], AP2 [3.4.2], AP3 [3.4.3]).
2. Input a test ANLDE system (defined by a user or generated automatically) (AU1 [3.2.1], AU3 [3.2.3], AS1 [3.3.1]).
3. Send the test ANLDE system to the server (AU3 [3.2.3], AP1 [3.4.1], AP3 [3.4.3]).

4. Wait for the report on solution (AS4 [3.3.4], AS5 [3.3.5], AU3 [3.2.3], AS1 [3.3.1], AP1 [3.4.1], AP2 [3.4.2], AP3 [3.4.3]).

5. View the report on solution (F1b [3.1.2], AU1 [3.2.1], AU2 [3.2.1], AU3 [3.2.3], AS1 [3.3.1], AP1 [3.4.1], AP2 [3.4.2]).

Rationale

This is the primary user activity (Fig. 2). The requirement extends Req. F1a [3.1.1].

References

System requirements: inputANLDESystem, sect. 7.6; generateANLDESystem, sect. 7.8; sendANLDESystem, sect. 7.10; solveANLDESystem, sect. 7.4; sendProcessMessage, sect. 7.21; sendANLDESystemReport, sect. 7.22; loadANLDESystems, sect. 7.3.
Use cases: Process a test ANLDE system, sect 8.2.

6.3 EU1b: Solving the set of ANLDE systems

Description

The web system solves set of ANLDE systems and produces the report on solution.

User Actions

1. Start the ANLDE processing subsystem (AU3 [3.2.3], AP1 [3.4.1], AP2 [3.4.2], AP3 [3.4.3], AD1 [3.5.1]).

2. Input a set of ANLDE systems (generated automatically or defined with user's text file). Solver and generator selection must be needed (AU1 [3.2.1], AU3 [3.2.3]).

3. Send the set to the server (AU3 [3.2.3], AP1 [3.4.1], AP3 [3.4.3]).

4. Wait for the report on solution (AS4 [3.3.4], AS5 [3.3.5], AU3 [3.2.3], AS1 [3.3.1], AP1 [3.4.1], AP2 [3.4.2], AP3 [3.4.3]).

5. View the report on solution (F2b [3.1.4], AU1 [3.2.1], AU2 [3.2.2], AU3 [3.2.3], AS1 [3.3.1], AP1 [3.4.1], AP2 [3.4.2]).

Rationale

This is the primary user activity (Fig. 2). The requirement extends Req. F2a [3.1.3].

References

System requirements: generateANLDESystemSet, sect. 7.9; saveANLDESystems, sect. 7.7; sendANLDESystemSet, sect. 7.11; solveANLDESystemSet, sect. 7.5; sendProcessMessage, sect. 7.21; sendANLDESystemSetReport, sect. 7.23; loadANLDESystems, sect. 7.3.
Use cases: Process a set of ANLDE systems, sect 8.3.

6.4 EU2a: User registration

Description

The web system allows user to register when she/he wishes. The registered user has an unique identifier (nick name).

User Actions

1. Start the registration subsystem (AU3 [3.2.3], AD1 [3.5.1]).

2. Fill the form (AU3 [3.2.3]).

3. Send the contents of the form to the server (AU3 [3.2.3]).

4. Wait for reply from the server (AU3 [3.2.3], AP3 [3.4.3]).

5. Get the acknowledgment (AU3 [3.2.3]).

Rationale

This is the secondary user activity (Fig. 2). The requirement extends Req. F4 [3.1.6] and Req. F5 [3.1.7]. It's necessary for more detalied analysis of user's activity.

References

System requirements: registerUser, sect. 7.13; logInUser, sect. 7.14; sendAcknowledgments, sect. 7.20.
Use cases: Register a user, sect 8.6. Log In, sect 8.4.

6.5 EU2b: User notes

Description

The web system allows user to send her/his opinion on the solution result. A special case here is user's disagreement with found solution(s) of the processed ANLDE system.

User Actions

1. Make decision on a note type:
   (a) a note about the web system (as a whole).
   (b) a note with the processed ANLDE system (e.g. for disagreement or commenting).
   (c) an agreement notification.
2. Start the note subsystem [1a, 1b] (AU3 [3.2.3], AD1 [3.5.1]).
3. Compose message [1a, 1b] (AU3 [3.2.3]).
4. Send the message [1a, 1b] (AU3 [3.2.3]).
5. Wait for reply from the server (AU3 [3.2.3], AP3 [3.4.3]).
6. Get reply from the server (AU3 [3.2.3]).

Rationale

This is the secondary user activity (Fig. 2). The requirement extends Req. F3 [3.1.5].

References

System requirements: sendUserNotes, sect. 7.12; sendAcknowledgments, sect. 7.20.
Use cases: Send a note, sect 8.5.

6.6 EU2c: Regular User Limits Management

Description

A regular user may manage her/his own limits on the solution process.

User Actions

(Regular user)

1. Start limits management (AU3 [3.2.3], AD1 [3.5.1], AS2 [3.3.2]).
2. Receive the form (AP2 [3.4.2], AP3 [3.4.3]).
3. Change his/her own limits.
4. Confirm changes (AP2 [3.4.2]).
5. Receive acknowledgment (AP3 [3.4.3]).

Rationale

This is the secondary user activity (Fig. 2). The requirement extends Req. AS4 [3.3.4].

References

System requirements: manageUserLimits, sect. 7.16; sendAcknowledgments, sect. 7.20.
Use cases: Manage user limits, sect 8.7.

6.7 EU2d: Log In

Description

The web system allows user to log in.

User Actions

1. Receive the form (AU3 [3.2.3], AD1 [3.5.1], AS2 [3.3.2], AP2 [3.4.2], AP3 [3.4.3]).
2. Input login and password.
3. User logs in in case of correct input (AP1 [3.4.1], AP2 [3.4.2],).
4. Receive acknowledgment (AP3 [3.4.3]).

Rationale

This is the secondary user activity (Fig. 2). The requirement extends Req. F4 [3.1.6].

References

System requirements: logInUser, sect. 7.14; sendAcknowledgments, sect. 7.20.
Use cases: Log In, sect 8.4.

6.8 EU3a: User Management

Description

The web system allows administrator to change or remove user accounts.

User Actions

(Administrator only)

1. Start user management (AU3 [3.2.3], AD1 [3.5.1], AS2 [3.3.2]).
2. Enter a user nickname.
3. Receive the form (AP2 [3.4.2], AP3 [3.4.3]).
4. Edit user information.
5. Confirm changes or remove the account (AP2 [3.4.2]).
6. Receive acknowledgment (AP3 [3.4.3]).

Rationale

This is the user management and administration (Fig. 2). The requirement extends Req. F4 [3.1.6].

References

System requirements: manageUsers, sect. 7.17; sendAcknowledgments, sect. 7.20.
Use cases: Manage users, sect 8.8.

6.9 EU3b: Activity Statistics

Description

The web system computes user activity statistics (available for administrator only). The web system generates report on generated and solved ANLDE systems, resource consumption and user notes statistics. The report is realized as a table with 2 columns: the first column contains user identifiers (IP addresses for unregistered users, or nick names for registered users), the second column contains activity metrics (number of generated systems, input systems, solved systems, acknowledged systems, resources).

User Actions

(Administrator only)

1. Start statistics subsystem (AD1 [3.5.1], AS2 [3.3.2], AS3 [3.3.3]).
2. Choose report type (AU3 [3.2.3]).

   (a) Select activity domain (nick names/IP addresses).
   (b) Select activity metrics (number of generated systems, input systems, solved systems, acknowledged systems, resources).
3. Send the request to the server (AU3 [3.2.3]).
4. Wait for the statistics (AP2 [3.4.2], AP3 [3.4.3]).
5. Receive the statistics (AU3 [3.2.3]).

Rationale

This is the user management and administration (Fig. 2). The requirement extends Req. F5 [3.1.7] and Req. F6 [3.1.8].

References

System requirements: requestStatisticsReport, sect. 7.19; sendAcknowledgments, sect. 7.20. sendStatisticsReport, sect. 7.24.
Use cases: Get statistics, sect 8.10.

6.10 EU3c: Default Limits Management

Description

The web system allows administrator to change default limits on solution process.

User Actions

(Administrator only)

1. Start limits management (AU3 [3.2.3], AD1 [3.5.1], AS2 [3.3.2]).
2. Receive the form (AP2 [3.4.2], AP3 [3.4.3]).
3. Change default limits.
4. Confirm changes (AP2 [3.4.2]).
5. Receive acknowledgment (AP3 [3.4.3]).

Rationale

This is the user management and administration (Fig. 2). The requirement extends Req. AS5 [3.3.5].

References

System requirements: manageDefaultLimits, sect. 7.18; sendAcknowledgments, sect. 7.20.
Use cases: Manage default limits, sect 8.9.

7 System Requirements

These requirements describe in more detail the functional structure of the expanded user requirements (see sect. 6). They define key functions (operations) that are supported by the web system. In use cases analysis (sect. 8) the system requirements are used as operations.

By the reason of limited resources, only the requirements with priority "primary" must be implemented. The requirements with priority "secondary" should be implemented, if no serious difficulties appear. The team will try to implement the requirements with priority "optional" but nothing is guaranteed.

7.1 startSession

Description: A user may start a session with Web-SynDic.

Author: Kirill A. Kulakov

Input Data: User profile or IP adress.

Sources of Input Data: Function logInUser [7.14].

Output Data: User profile and user limits or IP adress and default limits.

Destinations of Output Data: Web server.

Function requires User profile for registered users or IP adrress for regular users.

Preconditions: None.

Postconditions: None.

Restrictions: Req. AU3 [3.2.3].

Side Effects: None.

Moot Points: None.

Risks: None.

Priority: Primary.

7.2 finishSession

Description: After work with web server, user finish a session.

Author: Kirill A. Kulakov

Input Data: None.

Sources of Input Data: None.

Output Data: User activity.

Destinations of Output Data: Function updateStatistics [7.15].

Function requires None.

Preconditions: None.

Postconditions: None.

Restrictions: Req. AU3 [3.2.3].

Side Effects: None.

Moot Points: None.

Risks: User can close page or leave web system without sending any messages. Function updateStatistics [7.15] may be not implemented.

Priority: Primary.

7.3 loadANLDESystems

Description: This function loads ANLDE systems into the web system.

Author: Kirill A. Kulakov

Input Data: A test ANLDE system or set of ANLDE systems.

Sources of Input Data: Client.

Output Data: The test ANLDE system or the set of ANLDE systems.

Destinations of Output Data: Web system.

Function requires None.

Preconditions: The test ANLDE system or the set of ANLDE systems has been formed and function sendANLDESystem() or sendANLDESystemSet() has been started.

Postconditions: The test ANLDE system or the set of ANLDE systems may be sent to web server or not.

Restrictions: Req. AP1 [3.4.1], AP2 [3.4.2], AD1 [3.5.1], AU1 [3.2.1], AU3 [3.2.3], F1a [3.1.1], F2a [3.1.3].

Side Effects: None.

Moot Points: None.

Risks: User can sent wrong file.

Priority: Primary.

7.4 solveANLDESystem

Description: This function solves a test ANLDE system.

Author: Kirill A. Kulakov

Input Data: Test ANLDE system.

Sources of Input Data: Generator or function sendANLDESystem [7.10].

Output Data: Solver outcome.

Destinations of Output Data: Function sendANLDESystemReport [7.22].

Function requires User limits.

Pre-conditions: Session has been started. User limits must be set, otherwise the web system uses default limits.

Post-conditions: The solution may be found or not. It must be shown in solver outcome.

Restrictions: Req. AS4 [3.3.4], AS5 [3.3.5].

Side Effects: None.

Moot Points: None.

Risks: Solver can end with error message.

Priority: Primary.

7.5 solveANLDESystemSet

Description: This function solves set of ANLDE systems.

Author: Kirill A. Kulakov

Input Data: The set of ANLDE systems.

Sources of Input Data: Generator or function sendANLDESystemSet [7.11].

Output Data: Solver outcome.

Destinations of Output Data: Function sendANLDESystemSetReport [7.23].

Function requires User limits.

Preconditions: Session has been started. The set of ANLDE systems has been formed, loaded and sent to the web server.

Postconditions: The solution may be found or not.

Restrictions: Req. AS4 [3.3.4], AS5 [3.3.5].

Side Effects: None.

Moot Points: None.

Risks: Functions sendANLDESystemSetReport [7.23], solveANLDESystemSet [7.5] may be not implemented.

Priority: Optional.

7.6 inputANLDESystem

Description: This function inputs user's ANLDE system.

Author: Kirill A. Kulakov

Input Data: User limits.

Sources of Input Data: Client.

Output Data: ANLDE system.

Destinations of Output Data: Client.

Function requires User limits.

Preconditions: Client of the web system has been started.

Postconditions: None.

Restrictions: Req. AS5 [3.3.5].

Side Effects: None.

Moot Points: None.

Risks: User can input non ANLDE system.

Priority: Primary.

7.7 saveANLDESystems

Description: This function saves ANLDE systems with the user works.

Author: Kirill A. Kulakov

Input Data: ANLDE system or the set of ANLDE systems.

Sources of Input Data: Client.

Output Data: ANLDE system or the set of ANLDE systems in ANLDE format.

Destinations of Output Data: HTML or plain text file.

Function requires Selected file type: plain text or HTML.

Preconditions: User call web form to save ANLDE system or the set of ANLDE systems.

Postconditions: None.

Restrictions: Req. AU1 [3.2.1], AU2 [3.2.2].

Side Effects: None.

Moot Points: None.

Risks: It's possible that user didn't inputs or generates ANLDE systems and try to save.

Priority: Primary.

7.8 generateANLDESystem

Description: This function generates an ANLDE system.

Author: Kirill A. Kulakov

Input Data: Characteristics of the ANLDE system.

Sources of Input Data: User limits.

Output Data: ANLDE system.

Destinations of Output Data: Client.

Function requires None.

Preconditions: User sends query to generate ANLDE system.

Postconditions: None.

Restrictions: Req. AS1 [3.3.1], AS4 [3.3.4], AS5 [3.3.5].

Side Effects: None.

Moot Points: None.

Risks: Generator can end work with error message.

Priority: primary.

7.9 generateANLDESystemSet

Description: This function generates of a set of ANLDE systems.

Author: Kirill A. Kulakov

Input Data: Characteristics of the ANLDE systems.

Sources of Input Data: User limits.

Output Data: Set of ANLDE systems.

Destinations of Output Data: Client.

Function requires None.

Preconditions: User send query to generate set of ANLDE systems.

Postconditions: None.

Restrictions: Req. AS1 [3.3.1], AS4 [3.3.4], AS5 [3.3.5].

Side Effects: None.

Moot Points: None.

Risks: Generator can end work with error message.

Priority: Primary.

7.10 sendANLDESystem

Description: This function sends one ANLDE system to web server.

Author: Andrew Y. Salo

Input Data: ANLDE system.

Sources of Input Data: Function loadANLDESystems [7.3].

Output Data: ANLDE system.

Destinations of Output Data: Function solveANLDESystem [7.4].

Function requires None.

Preconditions: None.

Postconditions: None.

Restrictions: Req. AU3 [3.2.3], AP3 [3.4.3], AD1 [3.5.1].

Side Effects: None.

Moot Points: None.

Risks: None.

Priority: Primary.

7.11 sendANLDESystemSet

Description: This function sends set of ANLDE systems to web server.

Author: Andrew Y. Salo

Input Data: File in ANLDE format.

Sources of Input Data: Function loadANLDESystems [7.3].

Output Data: The set of ANLDE systems.

Destinations of Output Data: Function solveANLDESystemSet [7.5].

Function requires None.

Preconditions: None.

Postconditions: None.

Restrictions: Req. AU3 [3.2.3], AP3 [3.4.3], AD1 [3.5.1].

Side Effects: None.

Moot Points: None.

Risks: Function solveANLDESystemSet [7.5], sendANLDESystemSet [7.11] may be not implemented.

Priority: Optional.

7.12 sendUserNotes

Description: This function sends user notes to the server.

Author: Andrew Y. Salo

Input Data: Note (may contain additional data in format for an ANLDE system).

Sources of Input Data: Client.

Output Data: Note (may contain additional data in format for an ANLDE system).

Destinations of Output Data: Server.

Function requires Session must be started.

Preconditions: None.

Postconditions: Added note.

Restrictions: Req. AU3 [3.2.3], AP3 [3.4.3], AD1 [3.5.1].

Side Effects: None.

Moot Points: None.

Risks: Problems with storing notes.

Priority: Primary.

7.13 registerUser

Description: This function registers a new user.

Author: Andrew Y. Salo

Input Data: Nick name, password.

Sources of Input Data: Client.

Output Data: User profile.

Destinations of Output Data: Data store.

Function requires None.

Preconditions: None.

Postconditions: new registered user.

Restrictions: Req. AU3 [3.2.3], AP3 [3.4.3], AD1 [3.5.1].

Side Effects: None.

Moot Points: None.

Risks: Simultaneous registration of some users with identical nick names is possible.

Priority: Primary.

7.14 logInUser

Description: This function logs in a registered user.

Author: Andrew Y. Salo

Input Data: Nick name, password.

Sources of Input Data: Client.

Output Data: User profile in case of successful registration, IP adress otherwise.

Destinations of Output Data: Function startSession [7.1].

Function requires None.

Preconditions: None.

Postconditions: Registered user logged in.

Restrictions: Req. AU3 [3.2.3], AP3 [3.4.3], AD1 [3.5.1].

Side Effects: None.

Moot Points: None.

Risks: User can try to break open web system and log in as administrator.

Priority: Primary.

7.15 updateStatistics

Description: After session is over server updates statistic information on data store.

Author: Kirill A. Kulakov

Input Data: User activity.

Sources of Input Data: Function finishSession [7.2].

Output Data: Activity statistics.

Destinations of Output Data: Data store.

Function requires User profile for registered users or IP adress for regular users.

Preconditions: None.

Postconditions: None.

Restrictions: Req. F5 [3.1.7], F6 [3.1.8].

Side Effects: None.

Moot Points: None.

Risks: Function updateStatistics [7.15] may be not implemented.

Priority: Secondary.

7.16 manageUserLimits

Description: This function manages regular user limits.

Author: Petr A. Semin

Input Data: User limits.

Sources of Input Data: Session.

Output Data: Confirmation about changes, full list of user limits.

Destinations of Output Data: The web system (server).

Function requires: Default limits.

Preconditions: User limits are set (may be, equal to default limits).

Postconditions:

User limits has been changed according to the user's values.

List of user limits has been sent to user.

Restrictions: Req. AS4 [3.3.4].

Side Effects: None.

Moot Points: None.

Risks:

Storing of user limits can be difficult during implementation stage.

Function required default limits and depends on their constrains.

Function manageUserLimits [7.16] may be not implemented.

Priority: Secondary.

7.17 manageUsers

Description: Administrator changes/removes user accounts.

Author: Mikhail A. Kryshen'

Input Data: User accounts information.

Sources of Input Data: Web system.

Output Data: Changed user accounts information.

Destinations of Output Data: Web system (server).

Function requires Data store.

Preconditions: None.

Postconditions: None.

Restrictions: Req. AS2 [3.3.2].

Side Effects: None.

Moot Points: None.

Risks: When user accounts changes or removes, this user may log in into web system.

Priority: Primary.

7.18 manageDefaultLimits

Description: Administrator manages the following limits on the solution process: maximum time, memory, absolute values of coefficients, maximum number of equations, unknowns, ANLDE systems in a test set, solutions in Hilbert basis.

Author: Mikhail A. Kryshen'

Input Data: Current default limits.

Sources of Input Data: Web server

Output Data: Changed default limits.

Destinations of Output Data: Web server

Function requires Configuration file of the web system.

Preconditions: None.

Postconditions: none.

Restrictions: Req. AS2 [3.3.2], AS5 [3.3.5].

Side Effects: None.

Moot Points: None.

Risks: After change of default limits, web server must change user limits if they conflict with default limits. In case of active sessions web server must send message to all sessions.

Priority: Primary.

7.19 requestStatisticsReport

Description: Compute activity statistics, prepare data for the report requested by the administrator.

Author: Mikhail A. Kryshen'

Input Data: Log records, administrator's request for the report.

Sources of Input Data: Web system (server), form for choosing report type.

Output Data: Table of values.

Destinations of Output Data: Function sendStatisticsReport [7.24].

Function requires Function sendStatisticsReport [7.24].

Preconditions: None.

Postconditions: None.

Restrictions: Req. AS3 [3.3.3].

Side Effects: None.

Moot Points: None.

Risks: Function sendStatisticsReport [7.24] may be not implemented.

Priority: Secondary.

7.20 sendAcknowledgments

Description: This function sends different types of acknowledgments.

Author: Andrew V. Ananin

Input Data: Information for outputting in acknowledgment message.

Sources of Input Data: Function sendANLDESystem [7.10], Function sendANLDESystemSet [7.11], Function registerUser [7.13], Function sendUserNotes [7.12], Function manageUserLimits [7.16], Function manageDefaultLimits [7.18], Function manageUsers [7.17].

Output Data: Acknowledgment message.

Destinations of Output Data: Client part (browser).

Function requires Client part (browser).

Preconditions: Function sendANLDESystem [7.10] must be executed, Function sendANLDESystemSet [7.11] must be executed, Function registerUser [7.13] must be executed, Function sendUserNotes [7.12] must be executed, Function manageUserLimits [7.16] must be executed, Function manageDefaultLimits [7.18] must be executed, Function manageUsers [7.17] must be executed.

Postconditions: None.

Restrictions: Req. F1a [3.1.1], F2a [3.1.3], F3 [3.1.5], F4 [3.1.6], AU3 [3.2.3], AP3 [3.4.3], AD1 [3.5.1].

Side Effects: None.

Moot Points: None.

Risks: Functions sendANLDESystemSet [7.11], manageUserLimits [7.16], sendAcknowledgments [7.20] may be not implemented.

Priority: Secondary.

7.21 sendProcessMessage

Description: This function sends message about solution process.

Author: Andrew V. Ananin

Input Data: Information about ANLDE system or about a set of ANLDE systems.

Sources of Input Data: Function sendANLDESystem [7.10], Function sendANLDESystem-Set [7.11].

Output Data: Process message.

Destinations of Output Data: Client.

Function requires Session must be started.

Preconditions: An ANLDE system or a set of ANLDE systems must be processed with solver.

Postconditions: Send report on solution.

Restrictions: Req. F1a [3.1.1], F2a [3.1.3], AU3 [3.2.3], AS4 [3.3.4], AS5 [3.3.5], AP3 [3.4.3], AD1 [3.5.1].

Side Effects: None.

Moot Points: None.

Risks: Functions sendANLDESystemSet [7.11], sendProcessMessage [7.21] may be not implemented.

Priority: Secondary.

7.22 sendANLDESystemReport

Description: This function sends report on solution of ANLDE system.

Author: Andrew V. Ananin

Input Data: Test ANLDE system, solution, solver efficiency, server characteristics.

Sources of Input Data: Function solveANLDESystem [7.4].

Output Data: Report on solution an ANLDE system.

Destinations of Output Data: Client.

Function requires Session must be started.

Preconditions: Function solveANLDESystem [7.4] must be executed.

Postconditions: None.

Restrictions: Req. F1a [3.1.1], F1b [3.1.2], AU1 [3.2.1], AU2 [3.2.2], AU3 [3.2.3], AP3 [3.4.3], AD1 [3.5.1].

Side Effects: None.

Moot Points: None.

Risks: Solver can end with error message.

Priority: Primary.

7.23 sendANLDESystemSetReport

Description: This function sends report on solution of set of ANLDE systems.

Author: Andrew V. Ananin

Input Data: Test ANLDE system, solution, solver efficiency, server characteristics.

Sources of Input Data: Function solveANLDESystemSet [7.5].

Output Data: Report on solution a set of ANLDE systems.

Destinations of Output Data: Client.

Function requires Session must be started.

Preconditions: Function solveANLDESystemSet [7.5] must be executed.

Postconditions: None.

Restrictions: Req. F2a [3.1.3], F2b [3.1.4], AU1 [3.2.1], AU2 [3.2.2], AU3 [3.2.3], AP3 [3.4.3], AD1 [3.5.1].

Side Effects: None.

Moot Points: None.

Risks: Functions solveANLDESystemSet [7.5] sendANLDESystemSetReport [7.23] may be not implemented.

Priority: Optional.

7.24 sendStatisticsReport

Description: This function sends report on user's statistics.

Author: Andrew V. Ananin

Input Data: Table of values.

Sources of Input Data: Function requestStatisticsReport [7.19].

Output Data: Report on table of values.

Destinations of Output Data: Client.

Function requires System administrator permissions.

Preconditions: Function requestStatisticsReport [7.19] must be executed.

Postconditions: None.

Restrictions: Req. F5 [3.1.7], F6 [3.1.8], AU2 [3.2.2], AU3 [3.2.3], AS2 [3.3.2], AS3 [3.3.3], AP3 [3.4.3], AD1 [3.5.1],

Side Effects: None.

Moot Points: None.

Risks: Functions requestStatisticsReport [7.19], sendStatisticsReport [7.24] may be not implemented.

Priority: Secondary.

8 Use cases

Use case model is the one of the most important views on the required web system functionality because it combines the expanded user requirements (sect. 6) and system requirements (sect. 7). Therefore, this is the most comprehensive model for the functions of the developing web system.

There are two main types of actors: a user and an external algorithm. The other actors are their descendants: regular user, registered user, and system administrator (users); solver and generator (external algorithms).

The identification of the use cases is mainly based on the expanded user requirements. Each expanded user requirement corresponds to one use case.

Analysis of each use case is based on the system requirements. They form all high-level operations of the web system (see the corresponding sequence diagrams).

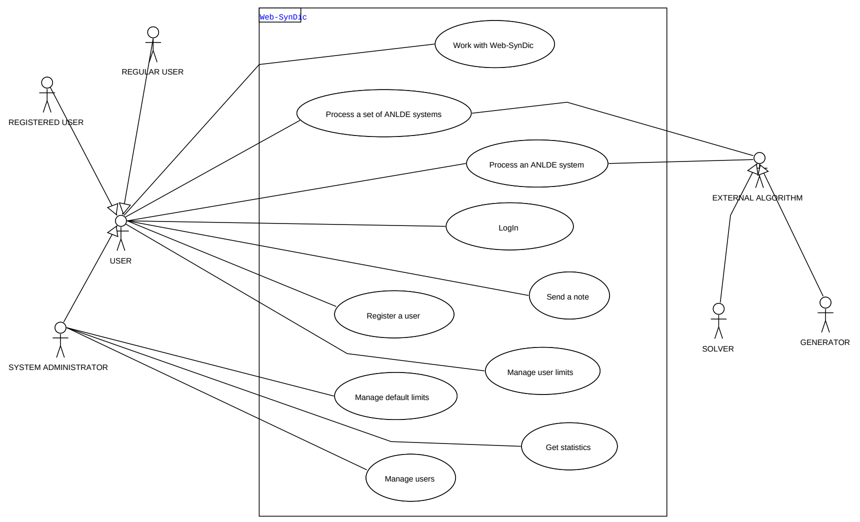

Figure 8: High-level use cases diagram

8.1 Work with Web-SynDic

8.1.1 Author

kirill A. Kulakov

Textual description

A regular user starts a session. The browser calls function `startSession()`. The web system creates the session. After work with the web system the user finishes the session. The browser calls function `finishSession()`. The web system destroys session and calls function `updateStatistics()` to update user activity.

8.1.2 High-level description

Use case	Work with Web-SynDic
Actors	User
Description	user's sessions.
References	Expanded user requirements: EU0 [6.1]. System requirements: startSession [7.1] (primary), finishSession [7.2] (primary), updateStatistics [7.15] (secondary).

8.1.3 Sequence diagram

See Figure 9.

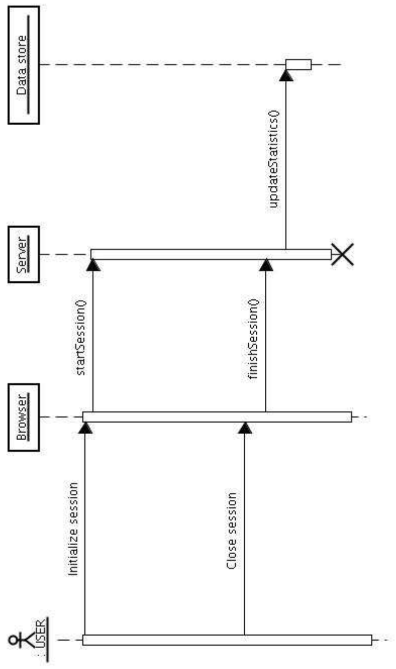

Figure 9: Sequence diagram for use case **Work with Web-SynDic**

8.2 Process an ANLDE system

8.2.1 Author

Kirill A. Kulakov

Textual description

An user initializes the subsystem to generate ANLDE system. The browser sends appropriate request to the web system. The web system calls the function generateANLDESystem [7.8]. Generator sends an ANLDE system to the web system, which sends form with ANLDE system to the browser.

An user initializes the subsystem to input ANLDE system manually. The browser sends appropriate request to the web system. The web system returns the form to input ANLDE system.

An user initializes the subsystem to solve ANLDE system. The user sends appropriate request to the browser. The browser calls function inputANLDESystem [7.6] to input ANLDE system into the web system. The web system calls function sendANLDESystem [7.10] to send the ANLDE system to the solver. When the solver works, the web system calls function sendProcessMessage [7.21] to send message about solution process to the browser. The solver calls function solveANLDESystem [7.4] and returns solution result to the web system. The web system calls function sendANLDESystemReport [7.22] to send the report on solution.

An user initializes the subsystem to save ANLDE system. The user sends appropriate request ANLDE system to browser. The browser calls function saveANLDESystems [7.7] to save ANLDE system.

8.2.2 High-level description

Use case	Process an ANLDE system
Actors	User, External algorithm
Description	Regular user sends an ANLDE system to solve. Solver gets ANLDE system from user or generator and solves it. Generator generates an ANLDE system.
References	User requirements: F1a [3.1.1], F1b [3.1.2]. Extended user requirements: EU1a [6.2]. System requirements: inputANLDESystem [7.6] (required), saveANLDESystems [7.7] (required), sendProcessMessage [7.21] (optional), sendANLDESystemReport [7.22] (required), sendANLDESystem [7.10] (required), generateANLDESystem [7.8] (reduction functionality), solveANLDESystem [7.4] (required).

8.2.3 Sequence diagram

See Figure 10.

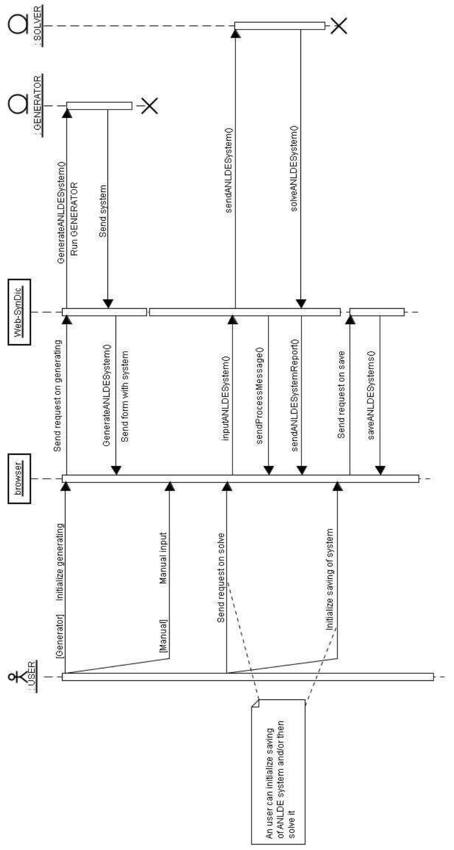

Figure 10: Sequence diagram for use case **Process an ANLDE system**

8.3 Process a set of ANLDE systems

8.3.1 Author

Kirill A. Kulakov

Textual description

An user initializes the subsystem to generate a set of ANLDE systems. The browser sends appropriate request to the web system. The web system calls function generateANLDESystemSet [7.9]. Generator sends an set of ANLDE systems to the web system and web system sends the form with an set of ANLDE systems to the browser.

An user initializes the subsystem to input ANLDE system from user's file. The browser sends appropriate request to the web system. The web system returns the form to choose a file.

An user initializes the subsystem to solve a set of ANLDE systems. The user sends appropriate request to the browser. The browser calls function inputANLDESystem [7.6] to input the set of ANLDE systems into the web system. The web system calls function sendANLDESystemSet [7.11] to send the set of ANLDE systems to the solver. When the solver works, the web system calls function sendProcessMessage [7.21] to send message about solution process to the browser. The solver calls function solveANLDESystemSet [7.5] and returns solution result to the web system. The web system calls function sendANLDESystemSetReport [7.23] to send the report on solution.

An user initializes the subsystem to save a set of ANLDE systems. The user sends appropriate to browser. The browser calls function saveANLDESystems [7.7] to save a set of ANLDE systems.

8.3.2 High-level description

Use case	Process a set of ANLDE systems
Actors	User, External algorithms
Description	Regular user sends a set of ANLDE systems to solve. Solver gets a set of ANLDE systems from regular user or generator and solves it. Generator generates a set of ANLDE system to user or solver.
References	User requirements: F2a [3.1.3], F2b [3.1.4]. Extended user requirements: EU1b [6.3]. System requirements: inputANLDESystem [7.6] (required), saveANLDESystems [7.7] (required), sendProcessMessage [7.21] (optional), sendANLDESystemSetReport [7.23] (required), sendANLDESystemSet [7.11] (required), generateANLDESystemSet [7.9] (reduction functionality), solveANLDESystemSet [7.5] (required).

8.3.3 Sequence diagram

See Figure 11.

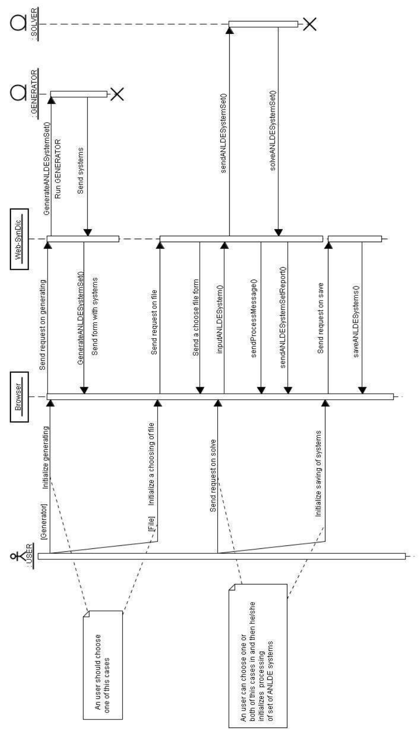

Figure 11: Sequence diagram for use case **Process a set of ANLDE systems**

8.4 Log In

8.4.1 Author

Andrey Y. Salo

Textual description

A regular user inputs his/her login and password in the form. The browser calls the function `loginUser()`. The web system tries to search for this user in data store. In case of success the

system checks his password. If the password is correct, the system changes the session and calls the function `sendAcknowledgment()`. From this moment the user acts as a registered user. If this user doesn't exist in the data store or the password is incorrect, the system calls the function `sendAcknowledgment()` and doesn't change the session. The user acts as a regular user.

8.4.2 High-level description

Use case	Log In
Actors	User
Description	User authentication.
References	User requirements: F4 [3.1.6]. Expanded user requirements: EU2d [6.7], EU3b [6.9]. System requirements: logInUser [7.14] (primary), manageUsers [7.17] (primary).

8.4.3 Sequence diagram

See Figure 12.

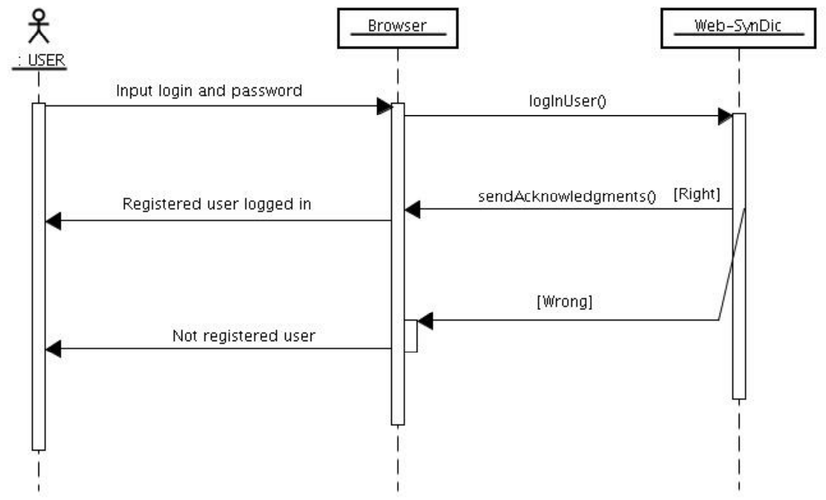

Figure 12: Sequence diagram for use case **Log In**

8.5 Send a note

8.5.1 Author:

Andrey Y. Salo

8.5.2 Textual description

An user initializes the subsystem for writing notes. The browser sends a request on note to the web system (note about the Web system, note with the processed ANLDE system). The web system returns appropriate form for writing note. User composes a message. The browser calls the function sendUserNotes(). The web system calls the function sendAcknowledgments().

8.5.3 High-level description

Use case	Send a note.
Actors	User.
Description	Message from the user.
References	User requirements: F3 [3.1.5]. Expanded user requirements: EU2b [6.5]. System requirements: sendUserNotes [7.12] (primary).

8.5.4 Sequence diagram

See Figure 13.

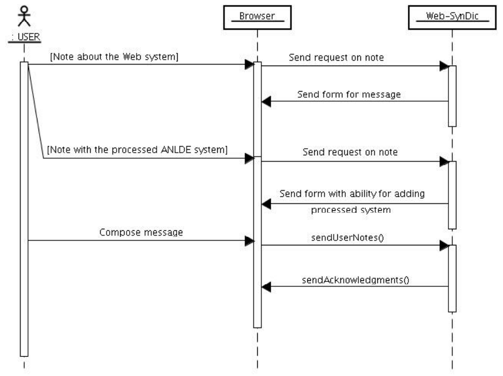

Figure 13: Sequence diagram for use case **Send a note**

8.6 Register a user

8.6.1 Author:

Andrey Y. Salo

Textual description

An user initializes the subsystem of registration. The browser sends request on a registration form. The web system returns the registration form. The user fills the registration form. The browser sends the filled form. The web system calls the function registerUser [7.13]. After that the web system calls function sendAcknowledgments [7.20] to send the results of registration.

8.6.2 High-level description

Use case	Register a user.
Actors	User.
Description	Registration of a user.
References	User requirements: F4 [3.1.6], F5 [3.1.7]. Expanded user requirements: EU2a [6.4], EU3a [6.8], EU3b [6.9]. System requirements: registerUser [7.13] (primary).

8.6.3 Sequence diagram

See Figure 14.

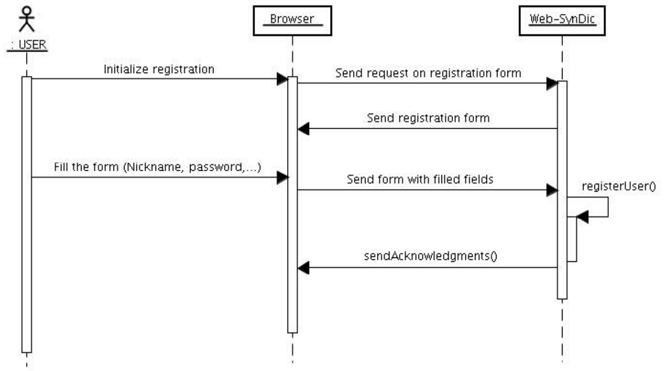

Figure 14: Sequence diagram for use case **Register a user**

8.7 Manage user limits

8.7.1 Author

Petr A. Semin

Textual description

Regular user initializes management of his/her limits by sending request for user limits form to server, using his/her browser. The server responds by sending one to user. User changes values of limits in the form, and sends this to the server. Server checks received values: if they exceed default limits, it sends message about invalid data to user; otherwise, it manages user's limits according to received data, and sends confirmation about changes to user.

8.7.2 High-level description

Use case	Manage user limits
Actors	User
Description	Regular user manages his/her limits (while not exceeding default limits)
References	User requirements: AS4 [3.3.4], AS5 [3.3.5]. Expanded user requirements: EU2c [6.6]. System requirements: manageUserLimits [7.16] (required), manageDefaultLimits [7.18]: (required).

8.7.3 Sequence diagram

See Figure 15.

8.8 Manage users

8.8.1 Author

Andrey V. Anan'in

Textual description

User management starts when user logs in as sysadmin and chooses management on the web page. The web system sends form for input a user's nickname. Then he inputs a nickname and sends the request on search. The web system searches this user in data store. If there are not corresponding user record in the data store, the web system sends report on absence this user. Otherwise the web system sends form with user's information. Sysadmin can change

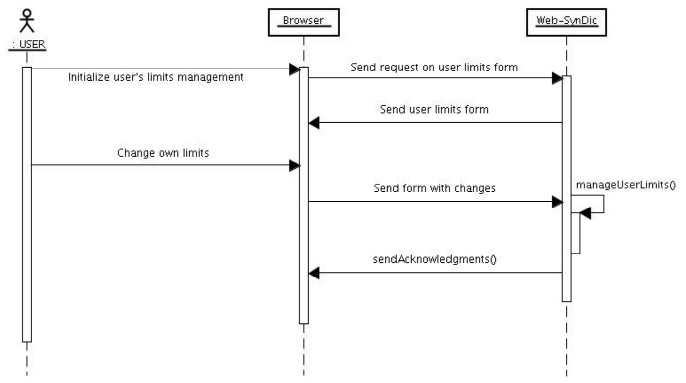

Figure 15: Sequence diagram for use case **Manage user limits**

this information or remove the user profile. Then the form is sent to the web system, and it performs requested operation and sends an acknowledgment to sysadmin.

8.8.2 High-level description

Use case	Manage users.
Actors	Sysadmin.
Description	Sysadmin can change or delete information about registered users.
References	User requirements: F4 [3.1.6], F5 [3.1.7], F6 [3.1.8], AS2 [3.3.2]. Expanded user requirements: EU2a [6.4], EU3b [6.9], EU3a [6.8]. System requirements: manageUsers [7.17] (primary), sendAcknowledgments [7.20](secondary).

8.8.3 Sequence diagram

See Figure 16.

8.9 Manage default limits

8.9.1 Author

Mikhail A. Kryshen'

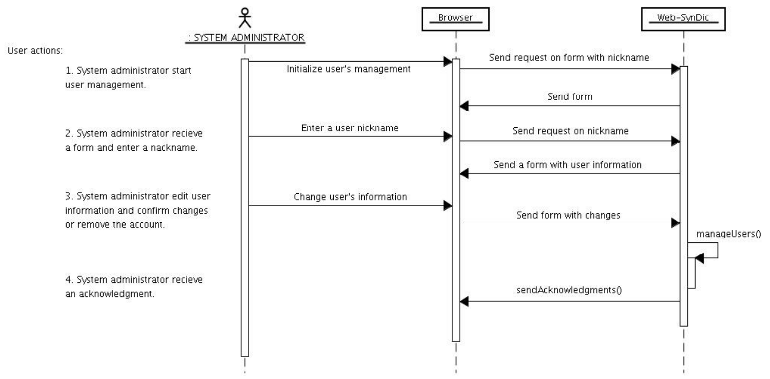
: SYSTEM ADMINISTRATOR
Browser
Web-SynDic
User actions:
1. System administrator start user management.
2. System administrator recieve a form and enter a nackname.
3. System administrator edit user information and confirm changes or remove the account.
4. System administrator recieve an acknowledgment.
Initialize user's management
Send request on form with nickname
Send form
Enter a user nickname
Send request on nickname
Send a form with user information
Change user's information
Send form with changes
manageUsers()
sendAcknowledgments()

Figure 16: Sequence diagram for use case **Manage users**

Textual description

Sysadmin initializes default limits management. Browser sends request to the web system. The web system sends limits management form filled with current values. Sysadmin edits values and sends request to the web system, which saves new values and sends acknowledgment to the sysadmin.

8.9.2 High-level description

Use case	Manage default limits
Actors	Sysadmin
Description	Sysadmin manages default system limits (user limits cannot exceed ones)
References	User requirements: AS4 [3.3.4]. Expanded user requirements: EU3c [6.10]. System requirements: manageDefaultLimits [7.18] (required).

8.9.3 Sequence diagram

See Figure 17.

8.10 Get statistics

8.10.1 Author

Mikhail A. Kryshen'

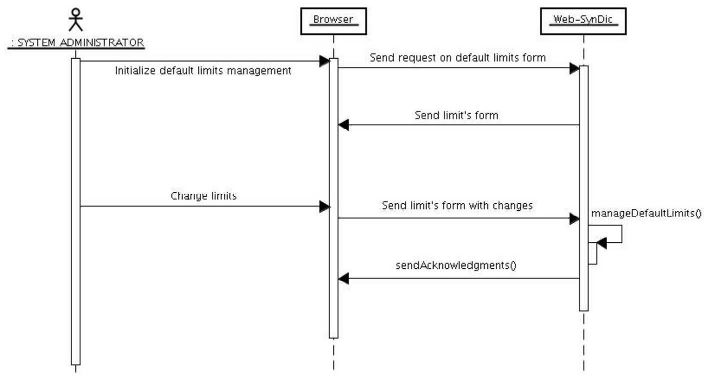

Figure 17: Sequence diagram for use case **Manage default limits**

Textual description

Sysadmin initializes activity statistics subsystem. Web system asks for the report type and parameters by sending a form to the client part. Sysadmin fills the form and requests the activity statistics report. Web system (server) generates the report and sends it to the client part.

8.10.2 High-level description

Use case	Get statistics
Actors	Admin
Description	Admin requests and recieves statistics report.
References	User requirements: F5 [3.1.7], F6 [3.1.8]. Expanded user requirements: EU3b [6.9]. System requirements: requestStatisticsReport [7.19] (required), sendStatisticsReport [7.24] (required).

8.10.3 Sequence diagram

See Figure 18.

9 Validation Criteria

Web-SynDic system is evaluated in the testing phase as per the requirements specification. The complete testing of a use case would be done through a testing phase, corresponding test case

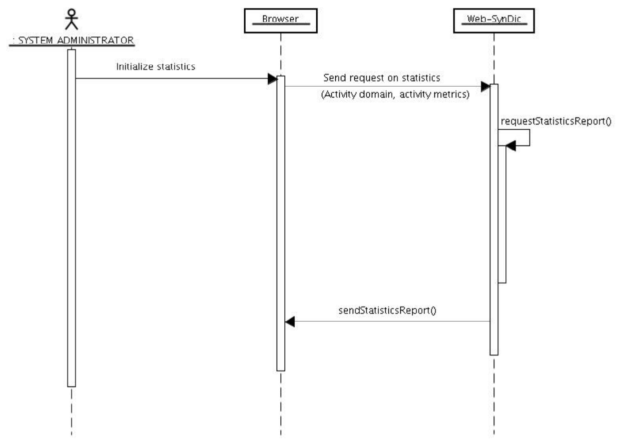
: SYSTEM ADMINISTRATOR
Browser
Web-SynDic
Initialize statistics
Send request on statistics
(Activity domain, activity metrics)
requestStatisticsReport()
sendStatisticsReport()

Figure 18: Sequence diagram for use case **Get statistics**

will be developed for each of them, and according to the Test Document (under construction).

Testing criteria for each use case is listed below.

It is assumed, that user always uses standard browser (see sect. 4.2.1) to interact with web system. There cannot be any other methods of interaction.

It is also assumed, that the user always provides valid data for requests of web system. Otherwise, web system does not process any invalid data in any context (does nothing with, or according to, it), except sending a brief information about mistake to the user.

Process an ANLDE system

When the user selects the single ANLDE system processing, he/she is presented with a page to feed in necessary ANLDE system that he/she wants to solve, or to set web system to generate one. The user fills in the information and requests for processing, afterward he/she will be provided with page containing source ANLDE system (entered or generated) and corresponding solution. If the processing takes more than 20 seconds, user receives progress notifications every 20 seconds until solution is done.

Following attributes are guaranteed to be satisfied with this use case: F1a [3.1.1], F1b [3.1.2].

Process a set of ANLDE systems

When the user selects the processing of a set of ANLDE system, he/she is presented with a page to feed in plain text file with ANLDE systems that he/she wants to solve, or to set web system to generate a set of ones. The user fills in the information and sends request

for solution process, afterward he/she will be provided with page containing source set of ANLDE system (entered or generated) and corresponding solutions. If the processing takes more than 20 seconds, user receives progress notifications every 20 seconds until solution is done.

Following attributes are guaranteed to be satisfied with this use case: F2a [3.1.3], F2b [3.1.4].

Log In

When previously registered user logs in successfully, web system changes the session, and provide the notification about successful login. From this moment the user acts as a registered user.

Following attributes are guaranteed to be satisfied with this use case: F4 [3.1.6].

Send a note

Users can interact with the Administrator by sending notes. When the user selects sending notes, he/she is presented with a page to feed in information, which she wants to send, and appropriate type of note (note about web system as a whole entity, opinion about particular ANLDE system, just agreements and approvals). Required data, i. e. ANLDE system, or personal data of registered user, is included automatically depending on selected type and user registration. The user fills in the information, sends it to web system and receives acknowledgment about successful delivery.

Following attributes are guaranteed to be satisfied with this use case: F3 [3.1.5].

Register a user

When user selects registration page of web system, he/she is provided with a page to feed in login, password and some personal information. The user fills the informations and sends request about actual registration. After login/password and some others checks, web system registers the user and creates corresponding profile in data store. From this moment, "Log In" use case becomes available for this user.

Following attributes are guaranteed to be satisfied with this use case: F4 [3.1.6], F5 [3.1.7].

Manage user limits

In this use case, the regular user selects the user limits management is presented with a page with a set of his/her current limits (may be, equal to default ones). Once the user fills in the requisite information and sends request for actual changes, web system will change his/her system limits, and return notification about successful changes.

Following attributes are guaranteed to be satisfied with this use case: AS4 [3.3.4], AS5 [3.3.5].

Manage users

In this use case, Administrator selects the user management and is initially presented with a request for a nick of registered user and a set of options to choose from (modify or remove user's data). Once Administrator fills in the requisite information and selects an option, web system will then provide him/her with a page to perform the management operation. Administrator edits the form as appropriate and submits it. If the operation is a successful, Administrator is returned notification that user's data has been successfully changed (removed).

Following attributes are guaranteed to be satisfied with this use case: F4 [3.1.6], F5 [3.1.7], F6 [3.1.8], AS2 [3.3.2].

Manage default limits

In this use case, Administrator selects the default limits management option and is presented with a page with a set of default limits. Once Administrator fills in the requisite information and sends request for actual changes, web system will change default system limits, and return notification about successful changes.

Following attributes are guaranteed to be satisfied with this use case: AS4 [3.3.4].

Get statistics

When Administrator selects the get statistics option, he/she will receive a page to make an initial request for report parameters (activity domain and activity metrics). Once these selections are made, web system will return a page containing the requested statistics. If the statistics evaluation takes more than 20 seconds, Administrator receives progress notifications every 20 seconds until evaluation is done.

Following attributes are guaranteed to be satisfied with this use case: F5 [3.1.7], F6 [3.1.8].

A Configuration Requirements

The web system is guaranteed to work in the following minimal configuration (other configurations may or may not work).

Server hardware (to be included in reports on solution according with requirements F1b.5 and F2b.4):

CPU: IA32, 1200 MHz;

RAM: 256 MB.

Server software: (to be included in reports on solution)

OS: Linux 2.4.19;

Java: Sun J2SDK 1.4.1, Apache Tomcat 4.

Client software:

Web browser: HTML 4.01 compatible.

References:

F1b.5 — Key hardware characteristics of the server must be included in the single ANLDE system solution outcome.

F2b.4 — Key hardware characteristics of the server must be included in the ANLDE systems set solution outcome.

AP1 — the web system must serve concurrently up to 5 users;

AP2 — the web system must not overload the server more than; 75% of the total server workload;

AP3 — the web system must reply on the user action less than after 20 seconds.

B External Algorithms

The following external algorithms are going to be used in Web-SynDic project (including demonstration and testing): the ANLDE solver (sect. B.1), slopes (sect. B.2), the ANLDE generator (sect. B.3), lp_solver (sect. B.4), and GLPK (sect. B.5). The bonsaiG algorithm was excluded from the list of the external algorithms due to problems with its installation.

B.1 The ANLDE solver

Short description: the ANLDE solver searches Hilbert basis of a homogeneous ANLDE system using the syntactic algorithms.

Input: ANLDE system (text file is used for input data). The first line in the file contains number of equations n and number of unknowns m. The next lines represent the given homogeneous ANLDE system by the corresponding CF-grammar (one line per rule): l_i is the index (started from zero) of the left-hand nonterminal of rule r_i ($i = 1, 2, \ldots, m$), a_{ki}

is the number of occurrences of symbol A_k in the right-hand side of rule r_i ($k = 1, 2, \dots, n$, $i = 1, 2, \dots, m$).

Input format	Math. format	Hom. ANLDE system
$n \quad m$ $l_1: \quad a_{11}\ a_{21}\ \dots\ a_{n1}$ $l_2: \quad a_{12}\ a_{22}\ \dots\ a_{n2}$ $\dots$ $l_m: \quad a_{1m}\ a_{2m}\ \dots\ a_{nm}$	$A_{k_1} \to A_1^{a_{11}} A_2^{a_{21}} \cdots A_n^{a_{n1}}$ $A_{k_2} \to A_1^{a_{12}} A_2^{a_{22}} \cdots A_n^{a_{n2}}$ $\dots$ $A_{k_m} \to A_1^{a_{1m}} A_2^{a_{2m}} \cdots A_n^{a_{nm}}$	$\sum_{k_i=1} x_i = \sum_{i=1}^{m} a_{2i} x_i$ $\sum_{k_i=2} x_i = \sum_{i=1}^{m} a_{2i} x_i$ $\dots$ $\sum_{k_i=n} x_i = \sum_{i=1}^{m} a_{ni} x_i$

Output: Hilbert basis, stdout is used for output.

```
There are q homogeneous solutions:
```
$h_1^{(1)}\ h_2^{(1)}\ \cdots\ h_m^{(1)}$
$h_1^{(2)}\ h_2^{(2)}\ \cdots\ h_m^{(2)}$
$\dots$
$h_1^{(q)}\ h_2^{(q)}\ \cdots\ h_m^{(q)}$

where q is the number of basis (minimal) solutions.

Specifics: the solver needs a lot of memory for solving large ANLDE systems; additional information, given by solver, will be ignored.

Purpose: in Web-SynDic the ANLDE solver is going to be directly used for searching Hilbert basis of homogeneous ANLDE systems (test ANLDE systems). This is the main demonstrated and tested algorithm.

Example:

Grammar	**ANLDE system**	**Input**	**Output**

$A \to AAB$
$A \to BB$
$B \to AAAB$
$B \to \varepsilon$
	$\begin{cases} x_1 + x_2 = 2x_1 + 3x_3 \\ x_3 + x_4 = x_1 + 2x_2 + x_3 \end{cases}$	`2 4`
		`0:2 1`
		`0:0 2`
		`1:3 1`
		`1:0 0`
			There are 2 homogeneous solutions:
			1 1 0 3
			0 3 1 6

B.2 Slopes

Short description: the slopes algorithm searches Hilbert basis for a homogeneous NLDE system.

Input: homogeneous NLDE system $Ax = \mathbb{O}$; stdin is used to read data.

$$\begin{array}{llll} n & m & & \\ a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ & & \cdots & \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{array}$$

Output: Hilbert basis; stdout is used to write data. See an example below.

Specifics: the slopes is licensed under GPL. The algorithm is not worth to use for large dimensions $n, m > 10 - 15$ for complexity reasons.

Purpose: in Web-SynDic the slopes is going to be used for searching Hilbert basis of a homogeneous test ANLDE system.

Example:

ANLDE system	2 4 Input	/* Output

$\begin{cases} x_1+x_2=2x_1+3x_3 \\ x_3+x_4=x_1+2x_2+x_3 \end{cases}$	`1 -1 3 0`
	`1 2 0 -1`
		`2 4`
		`1 -1 3 0`
		`1 2 0 -1`
		`*/`
		`Top (dimension 2):`
		
		`  0:  ( 1 1 1 1 ) { 1 2 } #=2`
		`     > 1 2`
		
		`Minimal support solutions:`
		
		`( 1 1 0 3 )`
		`( 0 3 1 6 )`
		
		`2 minimal support solutions`
		
		`Minimal solutions:`
		`( 1 1 0 3 )`
		`( 0 3 1 6 )`
		`2 minimal solutions`
		`Problem:  2x4, 3`
		`SlopesSyst-V3b:  No.  sols = 2,`
		`cputime = 0.000000/1 sec = 0.000000`

B.3 The ANLDE generator

Short description: the generator generates a test ANLDE system (in general case—with corresponding Hilbert basis) or a set of test ANLDE systems.

Input: type of generation method (T = 1, 2, or 3), number of systems to be generated ($N = 1, 2, \ldots$), and dimensions of systems (n equations, m unknowns), stdin is used for input.

$$\boxed{T \quad N \quad n \quad m}$$

Output: a generated set of test ANLDE systems, text file is used for output. Output file contains N generated systems. Each generated system is represented by one data section. The data sections are separated by empty lines.

The first line of a section contains number of unknowns m, number of equations n, and

number of solutions q. The second line contains vector I and the next n lines contain matrix A. The generated ANLDE system consists of n equations:

$$\sum_{i=1}^{m} \sigma_{ki} x_i = \sum_{j=1}^{m} A_{kj} x_j,\ k \in \overline{1,n}, \quad \text{where } \sigma_{ki} = \begin{cases} 1, & I_i = k \\ 0, & I_i \neq k \end{cases}$$

This ANLDE system can be also written in a compact way as $\mathrm{E}(I)x = Ax$.

The last m lines of the data section contain all minimal solutions $h^{(1)} \ldots h^{(q)}$ of the ANLDE system (its Hilbert basis).

The general format of a section is the following.

$$\boxed{\begin{array}{llll} m & n & q & \\ I_1 & I_2 & \ldots & I_m \\ A_{11} & A_{12} & \ldots & A_{1m} \\ A_{21} & A_{22} & \ldots & A_{2m} \\ \multicolumn{4}{c}{\ldots\ldots\ldots\ldots\ldots\ldots} \\ A_{n1} & A_{n2} & \ldots & A_{nm} \\ h_1^{(1)} & h_1^{(2)} & \ldots & h_1^{(q)} \\ h_2^{(1)} & h_2^{(2)} & \ldots & h_2^{(q)} \\ \multicolumn{4}{c}{\ldots\ldots\ldots\ldots\ldots\ldots} \\ h_m^{(1)} & h_m^{(2)} & \ldots & h_m^{(q)} \end{array}}$$

Specifics: several generation methods are available; not all ANLDE systems can be generated (potentially) using these methods. Generated systems can be degenerated (can be essentially simplified).

Purpose: in Web-SynDic the ANLDE generator is going to be used for automatic generation of test ANLDE systems whenever a user requires that (a single test ANLDE system or a set of them).

Example:

Description	Input	Output	ANLDE system

generation method $T = 2$;
$N = 2$ systems in the set;
each system with
$n = 2$ equations
and $m = 4$ unknowns
	`2 2 2 4`	`4 2 2`
		`1 2 2 2`
		`0 1 97 35`
		`0 0 1 35`
		`97 69`
		`0 34`
		`1 0`
		`0 1`
		
		`4 2 2`
		`1 2 2 1`
		`0 0 62 54`
		`0 0 73 21`
		`62 53`
		`72 21`
		`1 0`
		`0 1`
			System 1:
			$\begin{cases} x_1 = x_2 + 97x_3 + 35x_4 \\ x_2 + x_3 + x_4 = x_3 + 35x_4 \end{cases}$
			Hilbert basis:
			$(\ 97\ \ 0\ \ 1\ \ 0\ )$
			$(\ 69\ \ 34\ \ 0\ \ 1\ )$
			
			System 2:
			$\begin{cases} x_1 + x_4 = 62x_3 + 54x_4 \\ x_2 + x_3 = 73x_3 + 21x_4 \end{cases}$
			Hilbert basis:
			$(\ 65\ \ 72\ \ 1\ \ 0\ )$
			$(\ 53\ \ 21\ \ 0\ \ 1\ )$

B.4 lp_solver

Short description: lp_solver is based on the simplex algorithm; branch-and-bound is used for searching integer solutions (implemented as recursive function).

Input: specification of the MILP problem; stdin is used for input data. For ILP problems the following format is used:

$$\boxed{\begin{array}{l} \text{min: } c_1x_1 + c_2x_2 + \cdots + c_mx_m; \\ a_{11}x_1 + a_{12}x_2 + \cdots + a_{1m}x_m = b_1; \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2m}x_m = b_2; \\ \qquad\qquad \cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nm}x_m = b_n; \\ \text{int } x_1, x_2, \ldots, x_m; \end{array}}$$

Output: the optimal solution to the MILP problem; stdout is used for output.

$$\boxed{\begin{array}{ll} \multicolumn{2}{l}{\text{Value of objective function: } F} \\ \multicolumn{2}{l}{\text{Actual values of the variables:}} \\ x_1 & X_1 \\ x_2 & X_2 \\ \ \ \ldots & \\ x_m & X_m \end{array}}$$

where F, X_1, X_2, …, X_m are the searched optimal values.

Specifics: lp_solver is licensed under GPL; lp_solver is very time consuming for solving large systems ($n, m > 10 - 15$).

Purpose: in Web-SynDic lp_solver is going to be used for searching a particular solution of a test ANLDE system. This problem can be specified as ILP problem. Homogeneous ANLDE system forms the constraints $\mathrm{E}(I)x = Ax$; the additional constraint $\sum_{i=1}^{m} x_i \geq 1$ is used to eliminate the trivial solution $x = \mathbb{O}$; the objective linear function $f(x) = \sum_{i=1}^{m} c_i x_i$ is randomly generated (c_i are positive integers):

Example: Consider the following test ANLDE system

$$\begin{cases} x_1 + x_2 = 2x_1 + 3x_3 \\ x_3 + x_4 = x_1 + 2x_2 + x_3 \end{cases}$$

ILP problem	Input	Output

$x_1 + x_2 + 2x_4 \to \min$
$x_1 - x_2 + 3x_3 = 0$
$x_1 + 2x_2 - x_4 = 0$
$\sum x_i \geq 1,\ i \in \overline{1,4}$
$x_i \in \mathbb{Z},\ i \in \overline{1,4}$
	`min: x1+x2+2x4;`
	`x1-x2+3x3=0;`
	`x1+2x2-x4=0;`
	`x1+x2+x3+x4>=1;`
	
	`int x1,x2,x3,x4;`
		`Value of objective function: 8`
		
		`Actual values of the variables:`
		`x1          1`
		`x2          1`
		`x3          0`
		`x4          3`

B.5 GLPK

Short description: GLPK (GNU Linear Programming Kit) is a set of routines written in the ANSI C programming language and organized in the form of callable library. It is intended for solving linear programming (LP), mixed integer programming (MIP), and other related problems.

Input: specification of the problem, text file is used to read data. The CPLEX LP format is chosen for its simplicity (ILP problem):

```
Minimize f : c_1x_1 + c_2x_2 + ··· + c_mx_m
Subject To
Ax = b
Integer
x_1
x_2
...
x_m
End
```

Output: the optimal solution to the specified problem, text file is used for output. See an example below.

Specifics: GLPK is licensed under GPL; several input file formats are supported (MPS, CPLEX LP, GNU MathProg). GLPK is very time consuming for solving large systems ($n, m > 10 - 15$).

Purpose: in Web-SynDic GLPK is going to be used for searching a particular solution of a test ANLDE system. This problem can be specified as ILP problem. Homogeneous ANLDE system forms the constraints $\mathrm{E}(I)x = Ax$; the additional constraint $\sum_{i=1}^{m} x_i \geq 1$ is used to eliminate the trivial solution $x = \mathbb{O}$; the objective linear function $f(x) = \sum_{i=1}^{m} c_i x_i$ is randomly generated (c_i are positive integers):

```
Minimize f : c_1x_1 + c_2x_2 + ··· + c_mx_m
Subject To
(E(I) − A)x = 𝕆
x_1 + x_2 + ··· + x_m ≥ 1
Integer
x_1
x_2
...
x_m
End
```

Example:

ANLDE system	ILP problem	Input

$\begin{cases} x_1 + x_2 = 2x_1 + 3x_3 \\ x_3 + x_4 = x_1 + 2x_2 + x_3 \end{cases}$	$x_1 + x_2 + 2x_4 \to \min$
	$x_1 - x_2 + 3x_3 = 0$
	$x_1 + 2x_2 - x_4 = 0$
	$\sum x_i \geq 1,\ i \in \overline{1,4}$
	$x_i \in \mathbb{Z},\ i \in \overline{1,4}$
		`Minimize F: x1 + x2 + 2 x4`
		`Subject To`
		`x1 - x2 + 3 x3 = 0`
		`x1 + 2 x2 - x4 = 0`
		`x1 + x2 + x3 + x4 >= 1`
		`Integer`
		`x1`
		`x2`
		`x3`
		`x4`
		`End`

The output is the following.

```
Problem:  PROBLEM
Rows:  3
Columns:  4 (4 integer, 0 binary)
Non-zeros:  10
Status:  INTEGER OPTIMAL
Objective:  F = 8 (MINimum) 1.5 (LP)

No.  Row name Activity Lower bound Upper bound
------ ------------ ------------- ------------- -------------
1 r.4 0 0 =
2 r.5 0 0 =
3 r.6 5 1

No.  Column name Activity Lower bound Upper bound
------ ------------ ------------- ------------- -------------
1 x1 * 1 0
2 x2 * 1 0
3 x4 * 3 0
4 x3 * 0 0

End of output
```