Saint-Petersburg Russia

Microsoft Technologies in Theory and Practice of Programming

Contest – Conference

Software Requirements Specification

Project:	Web-SynDic
Customer:	Department of Computer Science, Head of the Department Yury A. Bogoyavlensky Petrozavodsk State University
Developers:	Dmitry G. Korzun (senior manager, PhD) Kirill A. Kulakov (technical manager, master student, BSc) Andrey Y. Salo (senior student) Andrey A. Ananin (junior student) Mikhail A. Kryshen (junior student)
Product:	Web System for Demonstrating and Testing the Syntactic Algorithms for Solving Linear Diophantine Equations in Nonnegative Integers

---

Contents

1 Introduction

The Web-SynDic project[1] is a student software engineering (SE) project of the Petrozavodsk State University (PetrSU), Department of Computer Science (CSDept).

The project is related to the research done at CSDept of PetrSU in development of a new type of algorithms for efficient solving some classes of nonnegative linear Diophantine equations (NLDE) by syntactic (parsing) methods [9, 10, 11]. These syntactic algorithms 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). For this class the syntactic algorithms allow efficient (polynomial and pseudo-polynomial) computations comparing with the general NLDE case when the same computational problems are NP-complete or even overNP [12].

The general goal of the project is to develop a full function web system[2] for visual demonstrating and testing the syntactic algorithms via the Internet. This allows researchers to input ANLDE systems (manually or automatically generated), search their Hilbert bases, test the correctness of the found solution, estimate the resource consumption, and compare the efficiency with available solvers, different from syntactic.

1.1 Purpose

In this document elicitated requirements to the anticipated software product are specified. The goal is to have a base for the design: collection and analysis of all requirements, fixation of acceptable in detail, and rejection/postpone of the others. The objectives are in constructing a model of the superposition: customer's needs at one side and project resource bounds at another side.

The document is intended for the customer and project development team. They all must accept it, then the document is frozen and any change must be requested, grounded and accepted by both sides.

All provisions of this document are the primary base and source for the subsequent phases of the project including the acceptance of the final product.

The more detailed analysis of the requirements can be found in the original Requirements Specification document [3].

1.2 Project overview

Nonnegative linear Diophantine equations are linear equations with integer coefficients and with solutions in nonnegative integers. NLDE and their systems are well-known area of mathematical

[1] The original, more detailed document set of Web-SynDic Project can be found at [1].

[2] The Web-SynDic Server is published at [2].

research, which goes back to Ancient Greece. Nowadays the algorithmic issues of this area are the nest of diverse topical problems in Mathematics and Computer Science.

An interesting case is NLDE systems associated with context-free (CF) grammars, so-called ANLDE systems [9]. The theory of ANLDE systems is a subject of recent research at the Department of Computer Science of the Petrozavodsk State University. This theory has roots in the work of Miguel Filgueiras and Ana-Paula Tomás [8] (Universidade do Porto, Portugal). Their results were later developed further by Dmitry Korzun in his PhD [10] (PetrSU, under scientific supervising of Dr. Yury Bogoyavlenskiy) with introducing new efficient polynomial algorithms for solving some classes of ANLDE-systems [11]. These algorithms are called "syntactic" according with their grammar-based nature.

Nowadays Web is the standard way to introduce recent results and progress to whom it may concern. This Project is intended to push forward the syntactic algorithms for comprehensive independent testing and to demonstrate for a wide spectrum of researches the computational efficiency of the algorithms. The anticipated web system should visually, on manually and automatically generated samples, present principal features of the syntactic algorithms, demonstrate to a user what computational behavior they have, and how efficiently they work comparing with available algorithms of other authors.

The developers of the Project are students of the CSDept of PetrSU. They study standard mandatory course "Software Engineering". The Project for them is a real problem exercise of applying SE techniques in practice.

The short name of the Project is **Web-SynDic**. This means that the software system to develop is **Web**-based; **SynDic** is the word **Syn***tact***ic** (the algorithms are syntactic), where tact is replaced with **D** according to our *tact*ful respect to Diophant as a father of the NLDE Theory. Based on formal grammars theory, one just uses the rule $tact \rightarrow \mathbf{D}$ to transform the string "Syntactic" to the string "SynDic".

1.3 Document conventions

Term	Description
Activity statistics	Data on user activity—how actively a user works with Web-SynDic.
ANLDE system	**A**ssociated with a formal grammar, NLDE system. See [9, 10].
AD#	Identifier for system constraint "**A**tribute: **D**eployment #".
AP#	Identifier for system constraint "**A**tribute: **P**erformance #".
AS#	Identifier for system constraint "**A**tribute: **S**ecurity #".
AU#	Identifier for system constraint "**A**tribute: **U**sability #".
CF-grammar	Context-free grammar

CS	Computer Science
CSDept	Computer Science Department. The PetrSU CSDept web-site is `http://www.cs.karelia.ru`
EU#	Expanded user requirement #.
F#	Identifier of the **F**unctional requirement #.
Hilbert basis	A set of all indecomposable (minimal) solutions of a homogeneous NLDE system.
G#	Identifier of the **G**eneral requirement # on the development process.
ILP	Integer linear programming
Indecomposable solution	A particular solution that is not a sum of two particular solutions.
lp_solve	The non-commercial linear programming code, written in ANSI C by Michel Berkelaar. Also it supports ILP problems. Available on `http://www.cs.sunysb.edu/~algorith/implement/lpsolve/implement.shtml`
NLDE	Nonnegative linear Diophantine equations, i.e. their solutions are in nonnegative integers and coefficients are integer. See for example [12, 13, 14].
Particular solution	Any non-trivial solution of a homogenous NLDE system.
PetrSU	Petrozavodsk State University, `http://petrsu.karelia.ru`
Report on solution	Outcome of solving process: solutions themselves and metrics for efficiency.
SE	Software Engineering, standard course books are [4, 5].
Slopes	Algorithm of M.Filgueiras and A.-P.Tomás for searching Hilbert basis of a homogenous NLDE system, available on `http://www.ncc.up.pt/~apt/dioph/`
Syntactic Algorithms	The algorithms that solve ANLDE system by constructing some derivations in the corresponding formal grammar, see [11, 10]. Web-SynDic is intended to demonstrate and test such algorithms.
Trivial solution	All-zero solution $\mathbb{0} = (0, \ldots, 0)$ of a homogeneous NLDE system.
UML	Unified Modelling Language. See for example [6, 7].
User note	Any user's opinion about Web-SynDic or about particular solution / generation process.
Web-SynDic	It stands for **Web**–based demonstrating and testing the **Syn**tact**ic** algorithms for solving nonnegative linear **D**iophantine equations.

1.4 References

[1] Web-SynDic Project development and maintenance site.
`http://zeta.cs.karelia.ru/Web-SynDic/doc/eng/` (English)
`http://zeta.cs.karelia.ru/Web-SynDic/doc/rus/` (Russian)

[2] The primary Web-SynDic Server is installed at
`http://zeta.cs.karelia.ru:8080/Web-SynDic/main.jsp`

[3] Web-SynDic: Requirements Specification. The original version.
`http://zeta.cs.karelia.ru/Web-SynDic/doc/eng/requirements/`

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

[5] Ian Sommerville. *Software Engineering.* 6th ed. Addison-Wesley, 2000.

[6] Graig Larman. *Applying UML and Patterns.* Prentice Hall, 2000.

[7] Jim Conallen. *Building Web Applications with UML.* Addison-Wesley, 2000.

[8] Miguel Filgueiras, 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.

[9] Yury A. Bogoyavlensky, Dmitry G. Korzun, *Obshchiy vid resheniya sistemy lineynih diophantovih uravneniy, associirovannoy s kontextno-svobodnoy grammatikoy.* Trudy Petrozavodskogo gosudarstvennogo universiteta. Ser. "Prikladnaya matematika i informatika". Vol. 6. Petrozavodsk, 1998. pp. 79–94. (in Russian)

[10] 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. (in Russian)

[11] 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.

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

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

[14] 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.

1.5 Document overview

This document is a part of the original Requirements Specification document [3], contains the elicitated user requirements and principal results of their analysis (use cases). The original Requirements Specification additionally presents in detail all phases of the requirements analysis and some supplementary results like validation criteria.

The elicitated user requirements are presented in Section 2 as a structured list with an identifier for each user requirement. The list is divided into three parts: general functions of the software (section 2.1), user characterization and corresponding constraints (section 2.2), general constraints and restrictions stated by the customer (section 2.3).

Specific user requirements for the software are presented in Section 3. These requirements are modeled with UML, based on use cases and they present the detailed view on what the web system must and should do from the functional point of view as well as system attributes and constraints on these functions.

2 Overall Description

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 (Hilbert basis or a particular solution) and some characteristics of the computation (time and memory consumption). 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 has an access to the Internet via a standard browser. This is enough for using the web system. The web system does not allow a user to have a direct access to the algorithms; it only shows an outcome of their work.

This general idea is expanded as a more detailed, structured list of general user requirements, presented in the next three subsections.

2.1 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 systems 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, BonsaiG and GLPK). 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.

2.2 User characterization

A regular Web-SynDic user is assumed to be a researcher in Diophantine analysis, formal grammars, integer programming, or related fields (or just a person with an interest in the area). No special knowledge in software engineering or networking is required.

Depending on user's interest the Web-SynDic system should support:

- solving a given ANLDE system or a set of them;
- generating an ANLDE system or a set of them, and then solving;
- estimating resource consumption (both time and space) of ANLDE solvers;
- testing the correctness of the syntactic algorithm on user's sample ANLDE systems;

- comparing outcomes and efficiency of the syntactic algorithm with alternative solvers.

Web-SynDic should also support 1) additional management for a privileged user via user registration, 2) limits on solution and generation processes, 3) activity statistics computation. Most of them are accessible only for a Web-SynDic system administrator.

2.3 General System Constraints

Web-SynDic is a web-based application. General system constraints for such applications typically includes deployment and performance issues.

Development requirements

G1: SE standards

The web system must be developed according with international principles, practice, standards and recommendations of SE. Use [4, 5, 6, 7] as basic books.

G2: Time resource

The Project starts on 16.07.2003 and the first working version has to be completed in November, 2003.

G3: Project languages

Working language of the Project is English. All artifacts should be replicated in Russian.

G4: Human resource

From the start of the project, students have to work at least 20 hours/week.

Deployment requirements

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.

AD2: Web-SynDic Server environment

The server part of Web-SynDic must be portable to majority of operation systems including Windows and UNIX families.

AD3: Portability to MS .NET

The software system should be portable to Microsoft .NET technology.

Performance requirements

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.

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.

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).

3 Specific Requirements

This section gives a detailed description of the requirements. This description is based on UML and use case models [6, 7].

The constructed use case model is one of the most important views on the required web system functionality. The model combines the expanded user requirements (EU) and system requirements, see the original SRS doc [3]. This is the resulting model for the functionality of the 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 defines a function and corresponds to one use case. The system requirements define relations inside the software system and correspond to high-level operations (see corresponding sequence diagrams).

3.1 Work with Web-SynDic

Author kirill A. Kulakov

Figure 1: High-level use cases diagram

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.

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

Sequence diagram is shown in Figure 2.

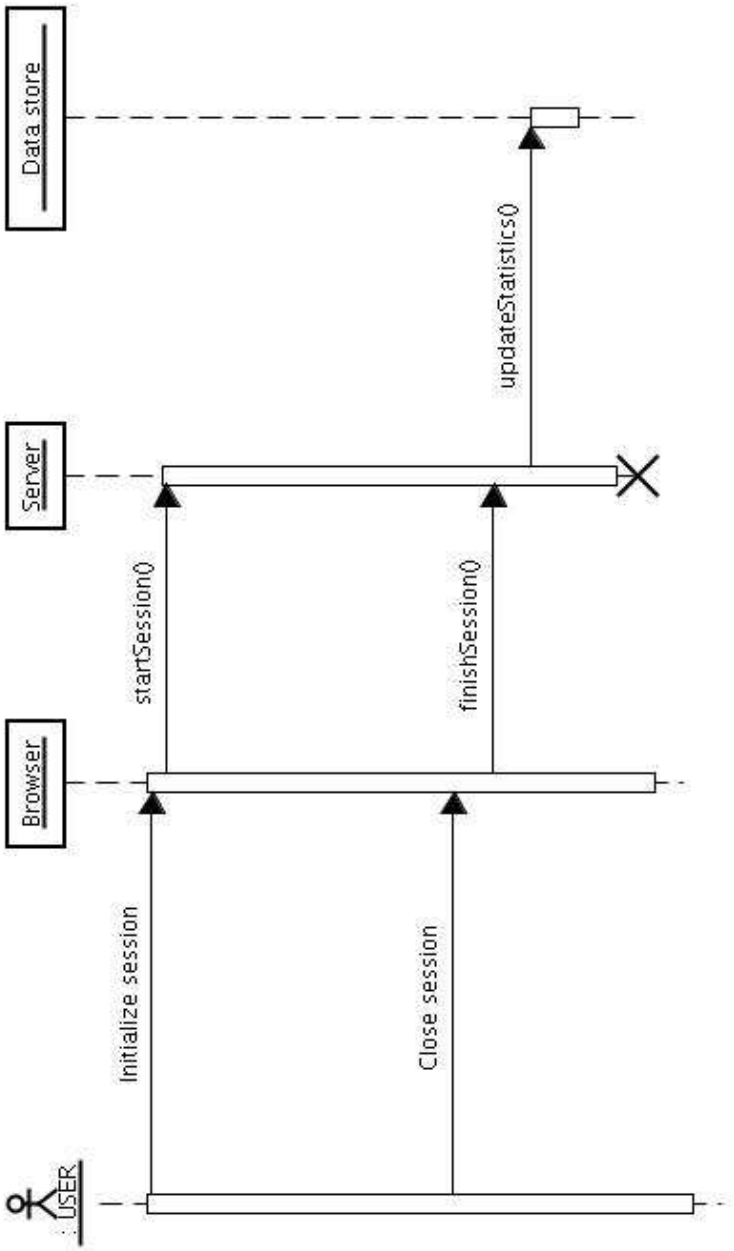

Figure 2: Sequence diagram for use case **Work with Web-SynDic**

3.2 Process an ANLDE system

Author Kirill A. Kulakov

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. 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 to input ANLDE system into the web system. The web system calls function sendANLDESystem to send the ANLDE system to the solver. When the solver works, the web system calls function sendProcessMessage to send message about solution process to the browser. The solver calls function solveANLDESystem and returns solution result to the web system. The web system calls function sendANLDESystemReport 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 to save ANLDE system.

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, F1b. Extended user requirements: EU1a. System requirements: inputANLDESystem (required), saveANLDESystems (required), sendProcessMessage (optional), sendANLDESystemReport (required), sendANLDESystem (required), generateANLDESystem (reduction functionality), solveANLDESystem (required).

Sequence diagram is shown in Figure 3.

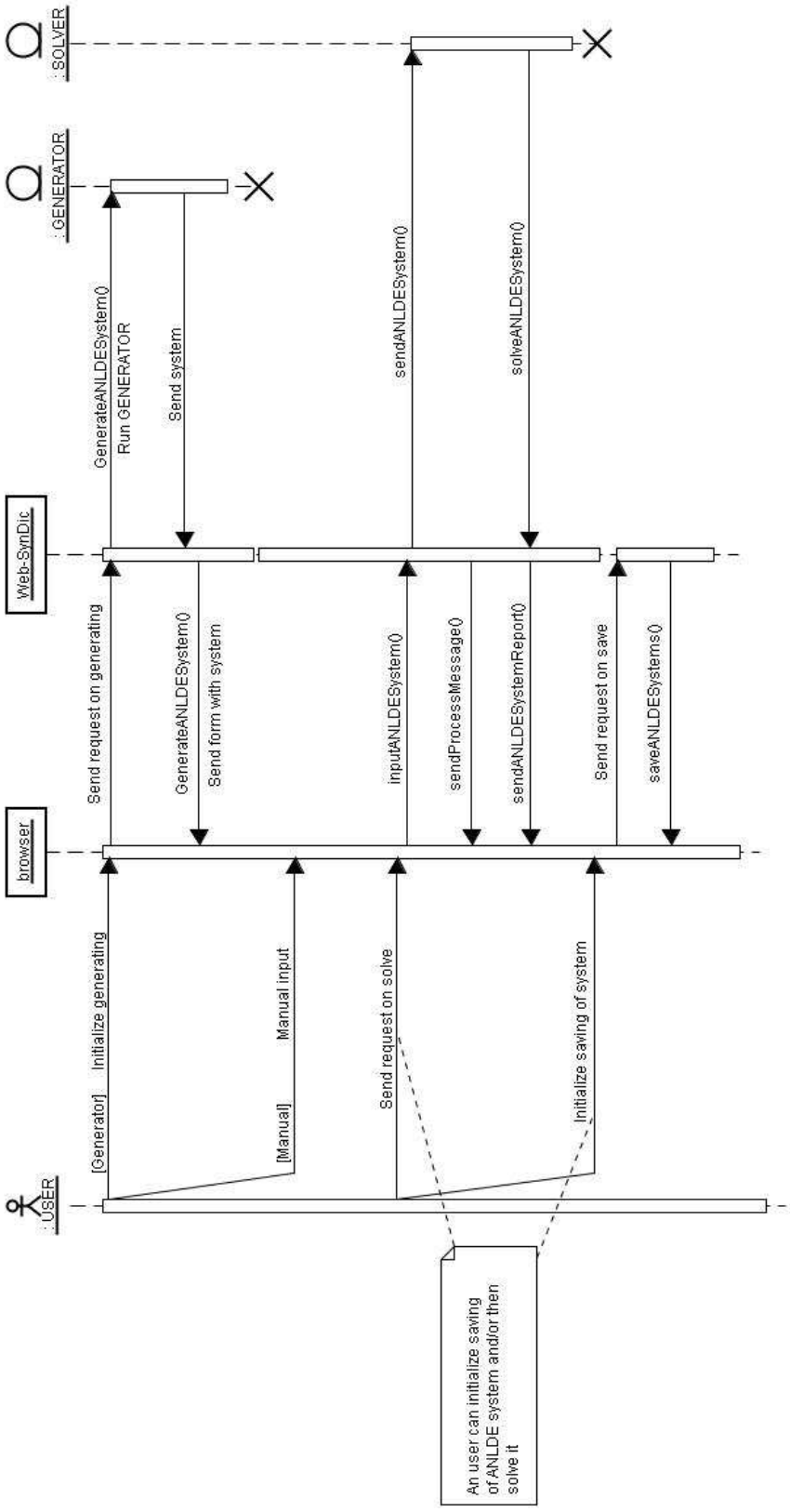

Figure 3: Sequence diagram for use case **Process an ANLDE system**

3.3 Process a set of ANLDE systems

Author Kirill A. Kulakov

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. 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 to input the set of ANLDE systems into the web system. The web system calls function sendANLDESystemSet to send the set of ANLDE systems to the solver. When the solver works, the web system calls function sendProcessMessage to send message about solution process to the browser. The solver calls function solveANLDESystemSet and returns solution result to the web system. The web system calls function sendANLDESystemSetReport 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 to save a set of ANLDE systems.

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, F2b. Extended user requirements: EU1b. System requirements: inputANLDESystem (required), saveANLDESystems (required), sendProcessMessage (optional), sendANLDESystemSetReport (required), sendANLDESystemSet (required), generateANLDESystemSet (reduction functionality), solveANLDESystemSet (required).

Sequence diagram is shown in Figure 4.

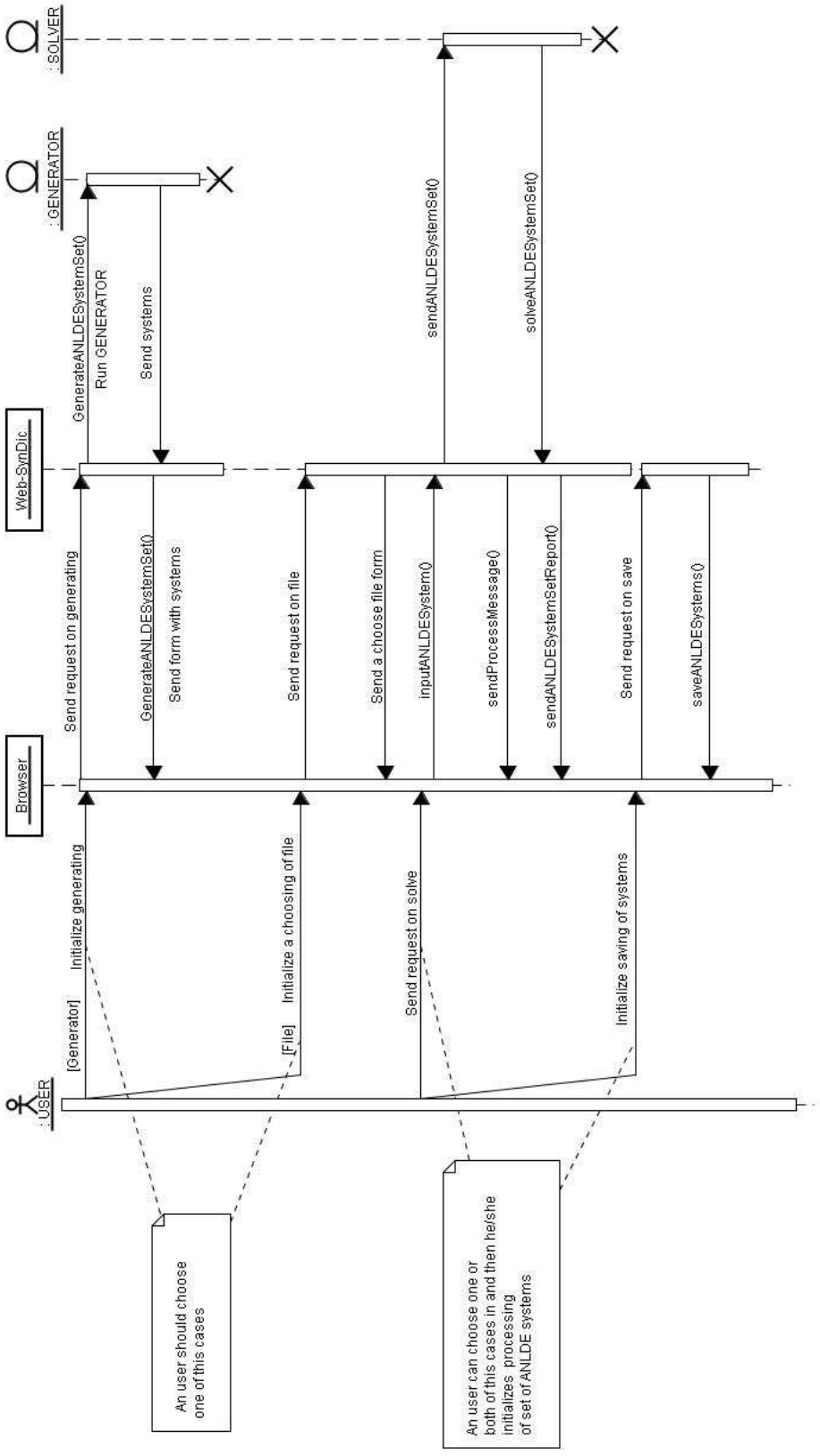

Figure 4: Sequence diagram for use case **Process a set of ANLDE systems**

3.4 Log In

Author Andrey Y. Salo

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.

Use case	Log In
Actors	User
Description	User authentication.
References	User requirements: F4. Expanded user requirements: EU2d, EU3b. System requirements: logInUser (primary), manageUsers (primary).

Sequence diagram is shown in Figure 5.

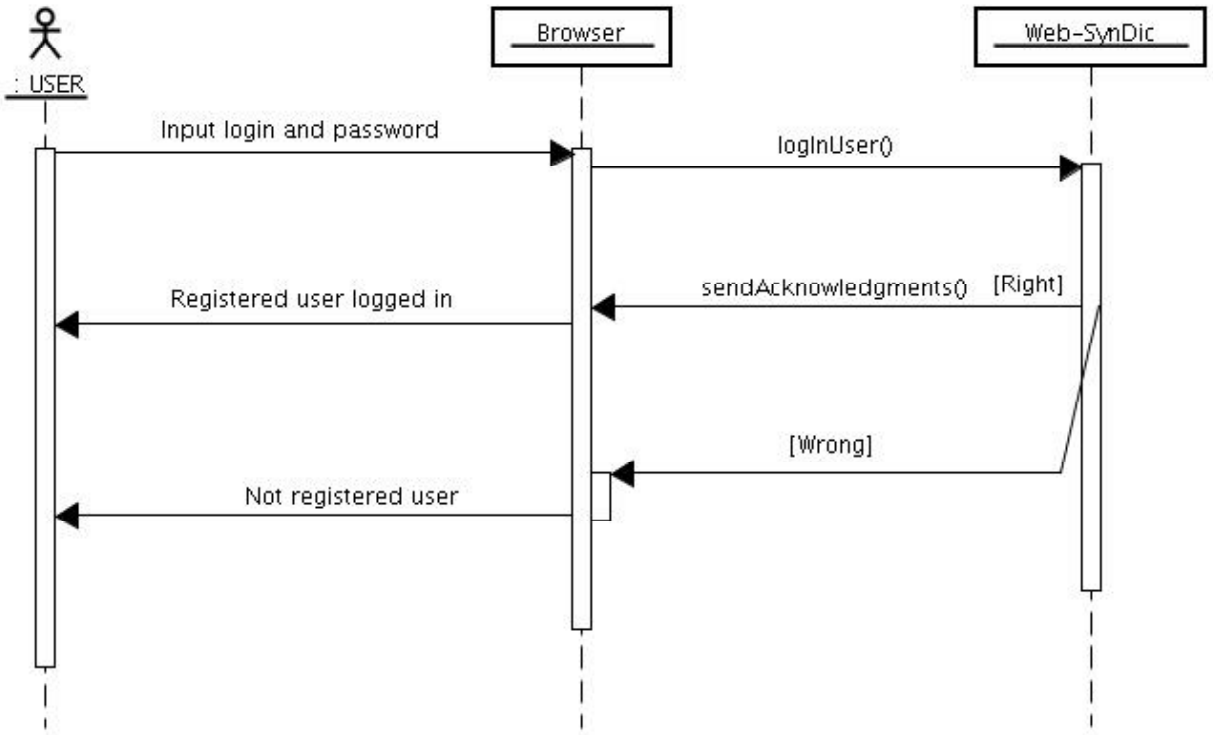

Figure 5: Sequence diagram for use case **Log In**

3.5 Send a note

Author Andrey Y. Salo

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().

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

Sequence diagram is shown in Figure 6.

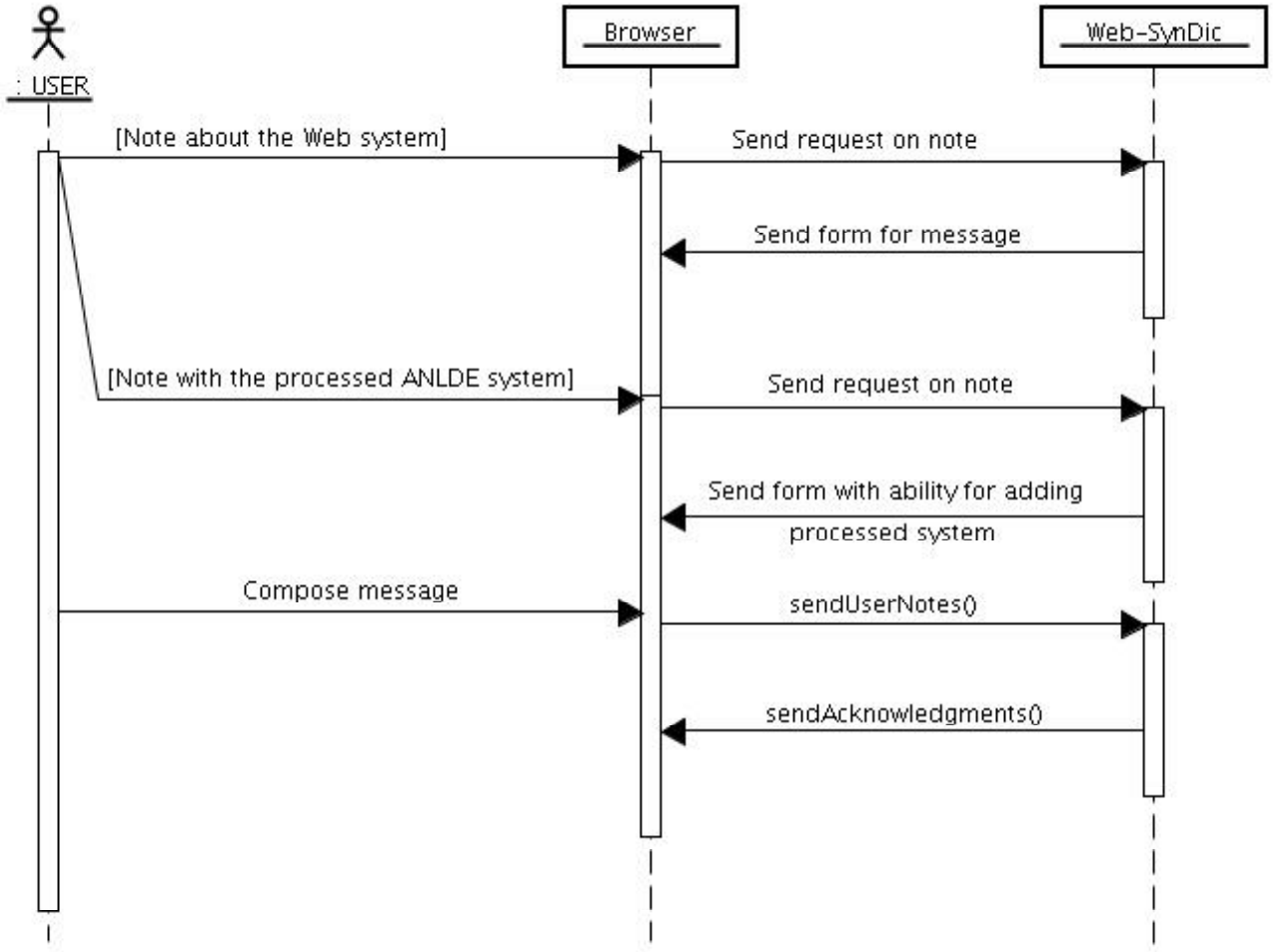

Figure 6: Sequence diagram for use case **Send a note**

3.6 Register a user

Author Andrey Y. Salo

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. After that the web system calls function sendAcknowledgments to send the results of registration.

Use case	Register a user.
Actors	User.
Description	Registration of a user.
References	User requirements: F4, F5. Expanded user requirements: EU2a, EU3a, EU3b. System requirements: registerUser (primary).

Sequence diagram is shown in Figure 7.

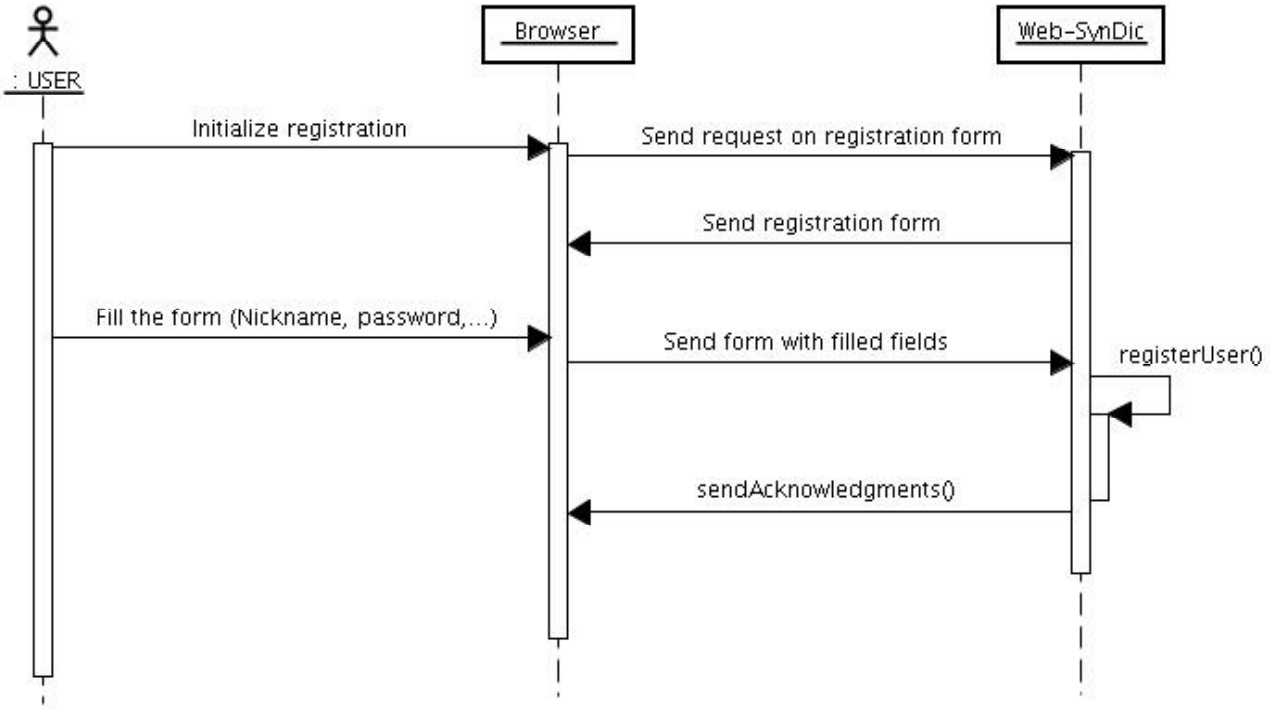

Figure 7: Sequence diagram for use case **Register a user**

3.7 Manage user limits

Author Petr A. Semin

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.

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

Sequence diagram is shown in Figure 8.

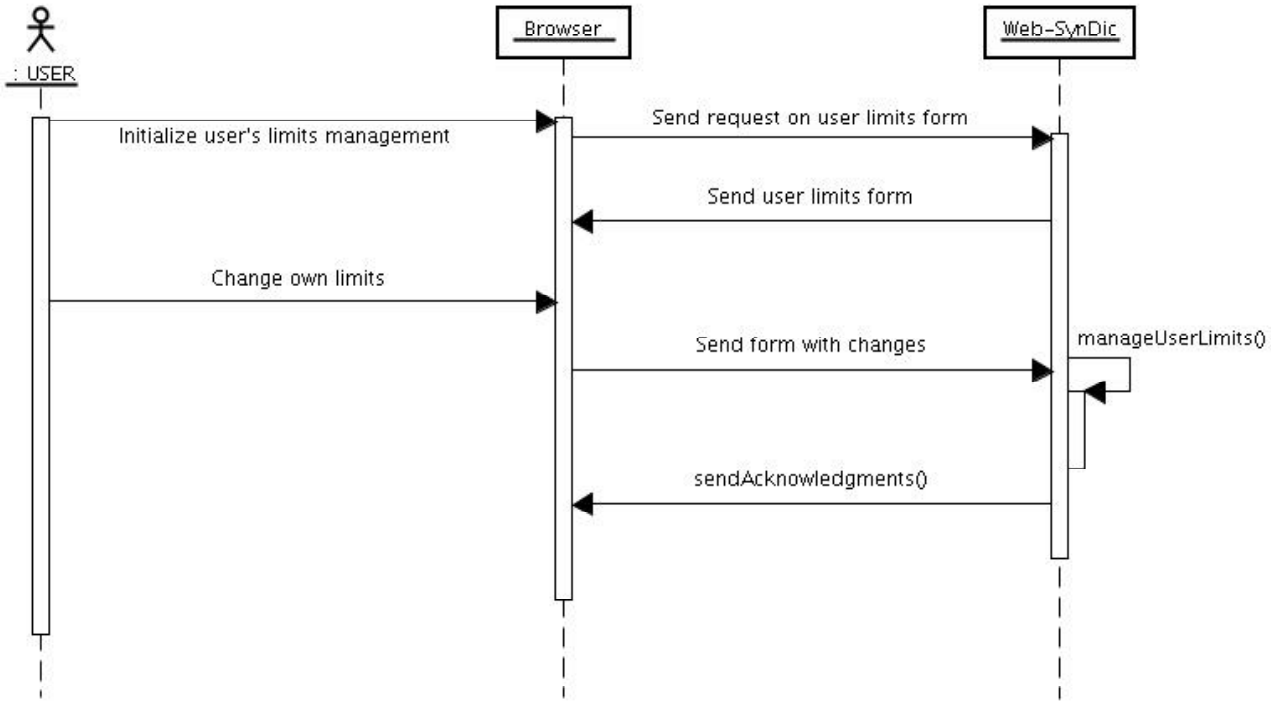

Figure 8: Sequence diagram for use case **Manage user limits**

3.8 Manage users

Author Andrey V. Anan'in

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 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.

Use case	Manage users.
Actors	Sysadmin.
Description	Sysadmin can change or delete information about registered users.
References	User requirements: F4, F5, F6, AS2. Expanded user requirements: EU2a, EU3b, EU3a. System requirements: manageUsers (primary), sendAcknowledgments (secondary).

Sequence diagram is shown in Figure 9.

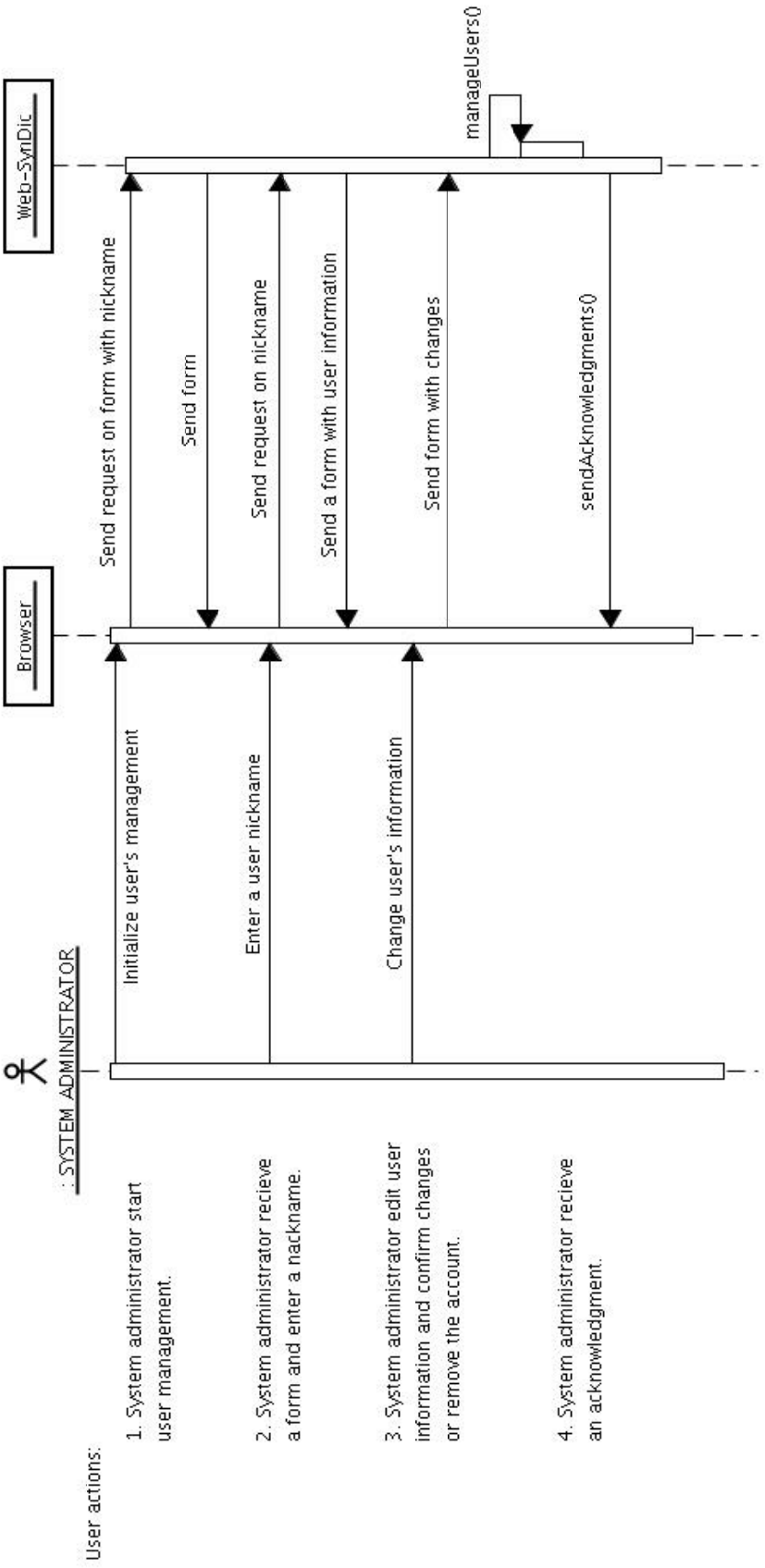

Figure 9: Sequence diagram for use case **Manage users**

3.9 Manage default limits

Author Mikhail A. Kryshen'

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.

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

Sequence diagram is shown in Figure 10.

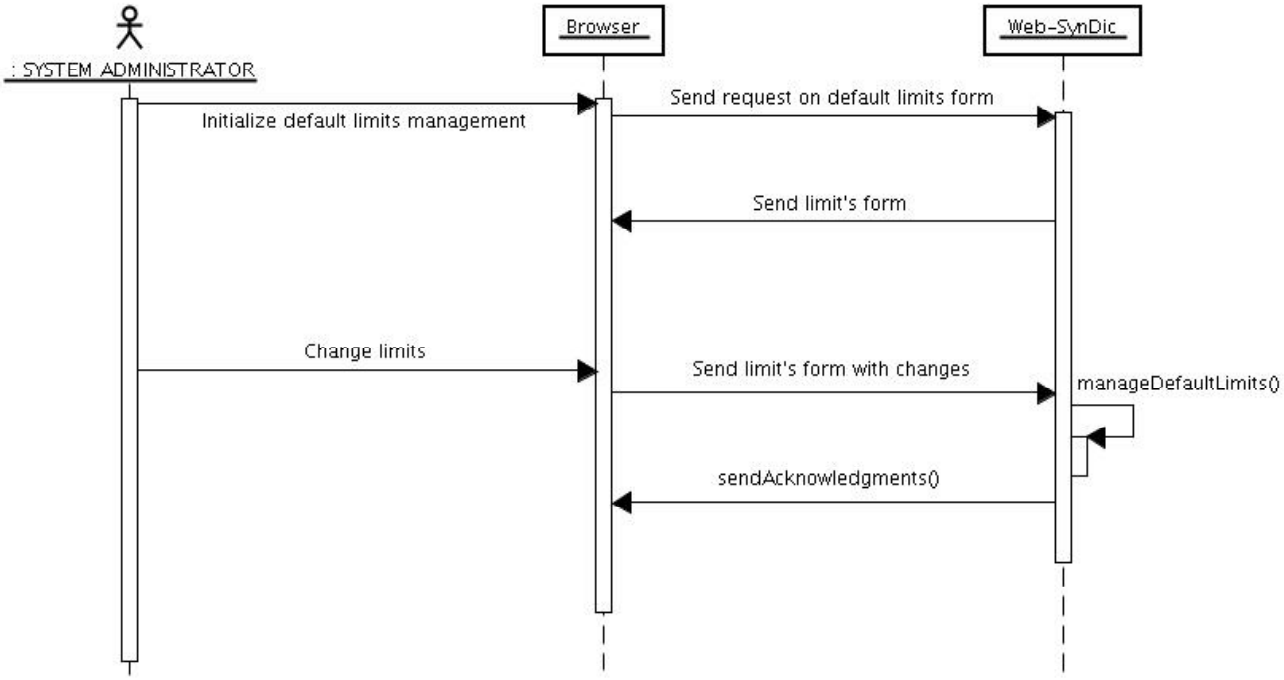

Figure 10: Sequence diagram for use case **Manage default limits**

3.10 Get statistics

Author Mikhail A. Kryshen'

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.

Use case	Get statistics
Actors	Admin
Description	Admin requests and receives statistics report.
References	User requirements: F5, F6. Expanded user requirements: EU3b. System requirements: requestStatisticsReport (required), sendStatisticsReport (required).

Sequence diagram is shown in Figure 11.

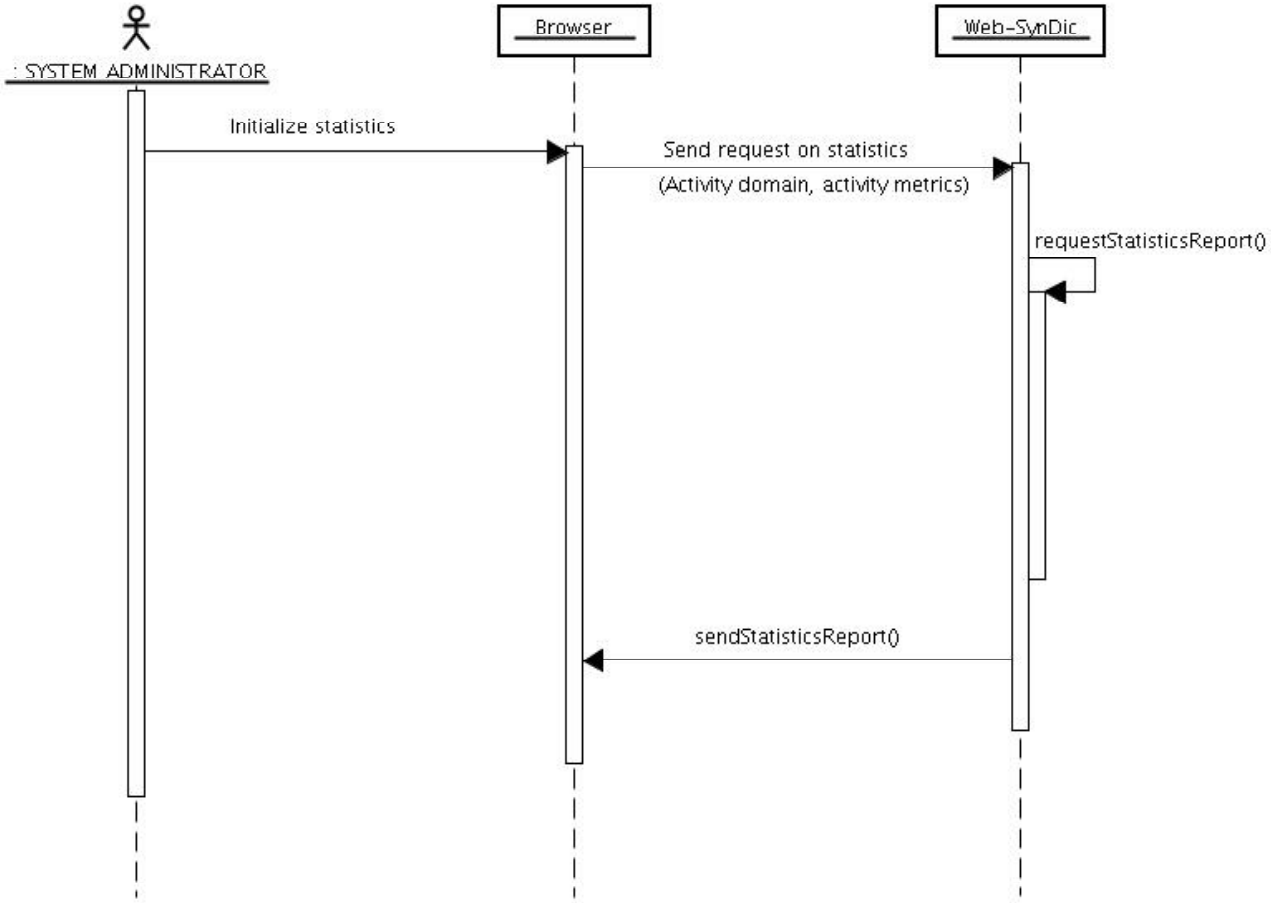

Figure 11: Sequence diagram for use case **Get statistics**