TALLINN UNIVERSITY OF TECHNOLOGY

Faculty of Information Technology

Department of Software Science

Olga Andrejeva, 176787 IAPM

TOOL FOR VISUALIZING GALOIS SUB-HIERARCHIES FOR MULTI-LAYERED SYSTEMS AT VARIOUS ABSTRACTION LEVELS

Master's thesis

Supervisor: Ants Torim, PhD

Tallinn 2019

TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Olga Andrejeva, 176787 IAPM

VAHEND GALOIS' ALAM-HIERARHIATE VISUALISEERIMISEKS ERINEVATEL ABSTRAKTSIOONIASTMETEL MITMEKIHILISTE SÜSTEEMIDE JAOKS

Magistritöö

Juhendaja: Ants Torim, PhD

Tallinn 2019

Author's declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Olga Andrejeva

07.05.2019

Tool for Visualizing Galois Sub-hierarchies for Multi-layered Systems at Various Abstraction Levels

Abstract

The primary goal of this thesis is to develop a Formal Concept Analysis (FCA) tool that would allow visualizing Galois sub-hierarchies (GSH) at different abstraction levels for multi-layered systems.

Formal concept analysis is widely used in different research fields, including IT. Representing FCA concepts and their hierarchies (relations) graphically allows to easily read and interpret the conceptual structure of the system under research. Galois sub-hierarchy graph does not contain empty concepts (the ones without objects and attributes), making it more clear. Even users untrained in FCA are showing high results in understanding visualized FCA graphs.

Today there is no proper GSH visualization tool for complex systems. Existing solutions that have basic GSH visualization functionality are outdated and have a number of known flaws. Support of relations between multiple layers and GSH generation on different abstraction levels for each context is important, as it allows to model and to analyze the conceptual structure of real systems.

The conducted study provides an overview of existing GSH visualization solutions and available technologies. The objectives and requirements are stated during the research and the solution design is developed based on the acquired knowledge. Implementation of the solution is compared to the existing GSH visualization tools and validated against real use case.

The main outcome of this work is designed and developed GSH visualization application that is capable of handling complex conceptual structures of multi-layered systems and generate visual output at different abstraction levels. The source code is made public in order to provide access to the tool to the greater auditory, to receive feedback from the greater number of people and to allow contributions.

This thesis is written in English and is 51 pages long, including 4 chapters, 24 figures and 3 tables.

Vahend Galois' alam-hierarhiate visualiseerimiseks erinevatel abstraktsiooniastmetel mitmekihiliste süsteemide jaoks

Annotatsioon

Käesoleva töö põhieemärk on arendata Formaalse Kontseptianalüüsi (FCA) vahend, mis võimaldaks visualiseerida Galois' alam-hierarhiad (GSH) mitmekihiliste süsteemidele erinevate abstraktsiooniastemetel.

Formaalne kontseptianalüüs on laialt kasutatav erinevates uurimisvaldkonnades, kaasa arvatud IT. FCA kontseptide ja nende hierarhiate (ehk vaheliste seoste) graafiline esitamine võimaldab uuritava süsteemi kontseptuaalsed struktuurid kergesti lugeda ja interpreteerida. Galois' alam-hierarhia graaf ei sisalda tühjad kontseptid (need mis ei sisalda objekte ega atribuute), seega graaf muutub kergemini arusaadavaks. Isegi kasutajad kellel puudub FCA kogemus näitavad kõrged tulemused FCA graafide arusaamises.

Tänapäeval puudub korralik GSH visualiseerimise vahend keeruliste süsteemide jaoks. Olemasolevad lahendused mis omavad GSH visualiseerimise põhifunktsionaalsust on vananenud ja omavad teatud puudused. Mitmekihiliste relatsioonide funktsionaalsuse toetamine ja GSH genereerimine erinevatel abstraktsiooniastemetel on oluline, kuna see võimaldab modelleerida ja analüüsida kontseptuaalse struktuuri reaalsest süsteemist.

Läbi viidud uuring pakub ülevaade olemasolevatest GSH visualiseerimise lahendustest ja saadavatest tehnoloogiatest. Eesmärgid ja nõuded olid määratud uuringu käigus ning lahenduse disain on arendatud saadud teadmiste põhjal. Lahenduse implementatsioon oli võrreldud olemasolevate GSH visualiseerimise vahenditega ning valideeritud reaalsete kasutusjuhtude vastu.

Töö põhitulemuseks on disainitud ja arendatud GSH visualiseerimise vahend mis võimaldab käsitleda mitmekihiliste süsteemide keerulised kontseptuaalsed struktuurid ning genereerida visuaalne väljund erinevate abstraktsiooniastmete jaoks. Lähtekood on avalikustatud, et vahend oleks kättesaadav suuremale kasutajate arvule, et saada tagasiside suuremast inimeste hulgast ja et lubada neile teha enda panuse koodi arendusele.

Lõputöö on kirjutatud Inglise keeles ning sisaldab teksti 51 leheküljel, 4 peatükki, 24 joonist, 3 tabelit.

List of abbreviations and terms

CSV (file)	*Comma-Separated Values* CSV file is a delimited text file that uses a coma to separate values
DOT	Graph description language
DS	Design science
DSRM	Design science research methodology
DTO	Data Transfer Object
FCA	Formal concept analysis
GSH	Galois sub-hierarchy
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IT	Information Technology
JAR (file)	*Java ARchive* JAR file is a package file (distribution) that aggregates Java classes and resources in it
NGD	Angular Dependencies Graph
npm	Node Package Manager
OS	Operating system
REST	Representational State Transfer
UX	User Experience
WAR (file)	Web Archive or Web Application Resource
XML (file)	*Extensible Markup Language* XML file is a text file which format is encoded by a set of rules defined by XML

Table of contents

List of Figures

List of Tables

Chapter 1

Introduction

The problem of data representation and analysis is of current interest for a lot of researches and specialists around the globe. Systems and models used in real life tend to be of a high complexity, thus representation requires particular strategies and methods. Among them is data analysis and knowledge representation method called Formal concept analysis (FCA), which is widely used in different research fields, such as linguistics [3], software engineering [4] [5] [6] [7] [8] [9], psychology, artificial intelligence (AI) [10] [11], and information retrieval [12].

FCA operates with ideas of concept and concept hierarchy and has solid mathematical foundations [13]. It makes it possible to handle conceptual structures mathematically, including constructing, representing and analyzing them [14]. One of the benefits of this approach is that it provides a possibility to represent conducted evaluations graphically, which gives us an overview of the objects and their relations in an easily readable form of a diagram. According to the study conducted in 2004, even users untrained in FCA are able to read and interpret that kind of diagrams [15], which means that it could be extremely helpful for conducting the analysis of systems, where conceptual structures are under research.

Graphical output of the FCA method may contain empty elements, which may be a problem for systems of high complexity, due to a greater number of elements presenting on the diagram, which decreases the readability and negatively influences on the outcome of an analysis. This situation can be improved by removing empty relations, which is possible if a Galois sub-hierarchy (GSH) of an original concept lattice is used.

The bigger the number of objects and attributes along with the density of relations, the harder is to conduct such kind of analysis without visualizing the structure. Unfortunately, today there is no proper visualization tool for creating diagrams for Galois sub-hierarchies: there is a lack of options for defining different layers and describing relations between them, as well as no proper solution for defining various abstraction levels of the layers (eg., atomic elements, packages, etc.) and sub-systems (which is usually the case in real life systems). The latter is very important due to the fact that GSH diagrams are very useful for defining, representing and analyzing sub-systems particularly.

The goal of this thesis is to create a solution for visualizing complex multi-layered conceptual lattices using Galois sub-hierarchies. Motivating use case for creation such tool is modelling and visualization of software systems [16]. The solution is designed based on existing knowledge. Output data provided by the tool under study will make further analyses with GSH involved more effective, save time and reduce the number of human mistakes, caused by interpretation of the complex data sets and hardened readability. The most complex the system under analysis (in terms of objects, attributes, and density of the relations), the most helpful the solution will be.

As a part of the validation process, the comparison of the implemented tool and existing solutions is conducted and obtained results are evaluated. The utility of the tool is demonstrated in a case study.

Thesis structure is as follows: *Chapter 1* defines the motivation and states the problem of the thesis as well as provides an overview of related existing knowledge. It also covers research design & methodology aspects. *Chapter 2* covers the design and the implementation of the solution, including technical details. *Chapter 3* describes the validation process and provides evaluations of the obtained results. Thesis outcome is summarized and presented in *Chapter 4*.

1.1 Related work

The initial base for the current research consists of two bachelors theses of the year 2012 created by Maarja Raud [17] and Kristo Aun [18], written under the supervision of Ants Torim, which results are implemented tools for visualizing Galois' sub-hierarchies using different approaches. Both works are fields for future improvements, but considering the technological changes during the years it was decided not to use the visualization part itself as a base for the current solution. However, after the conducted study, it was decided to partially use Maarja's code for converting concept lattice into GSH structure by adjusting it for the developed solution.

1.1.1 Theoretical background

FCA allows to hierarchically represent conceptual structures (found based on mathematical theory) by grouping them together in a concept lattice. Definitions of notions related to FCA are provided below.

Formal context is a triplet

$$K = (G, M, I)$$

where G is finite set of objects, M is finite set of attributes and I is a relation between them:

$$I \subseteq G \times M$$

Thus, a formal context is formed by the sets of formal objects and formal attributes together with their relation to each other and can be represented by a cross table [12] where table rows correspond to objects, table columns - to attributes and table entries represent relations between them (if particular object has particular attribute). The example of a formal context is provided in Table 1.1.

Formal concept - is a pair (A, B), where A is a set of formal objects (extension) and B is a set of formal attributes (intention), such that all formal objects in A share the attributes in B and all formal attributes in B are shared by formal objects in A [12]. Formal concepts derived based on the Table 1.1 are provided in Table 1.2.

	Attribute 1	Attribute 2	Attribute 3
Object 1	1	0	1
Object 2	0	0	1
Object 3	1	1	0

Table 1.1: Example of formal context - cross table representation

#	**Formal concept**
0	M = {}
	G = {Object 1, Object 2, Object3}
1	M = {Attribute 1, Attribute 2, Attribute 3}
	G = {}
2	M = {Attribute 1, Attribute 3}
	G = {Object 1}
3	M = {Attribute 3}
	G = {Object 1, Object 2}
4	M = {Attribute 1}
	G = {Object 1, Object 3}
5	M = {Attribute 1, Attribute 2}
	G = {Object 3}

Table 1.2: Formal concepts derived from Table 1.1

Sub-concept - is a formal concept, which contains an intent (set of formal attributes) of the super-concept. Dually, an extent (set of formal objects) of the sub-concept is contained in the super-concept (eg., concepts 4 and 5 from Table 1.2).

Concept lattice - hierarchically organized lattice, which consists of formal concepts and their sub-concepts [19], where concepts are represented as nodes and relations between concepts as edges. Used for visual representation of formal context, for example see Figure 1.1 which is a concept lattice of formal context presented in Table 1.1 (lattice is generated using Galicia v2.0 [20]).

Galois lattice - concept lattice, where each node is a formal concept and has two parts: the extension and the intention. The relations between them are described as

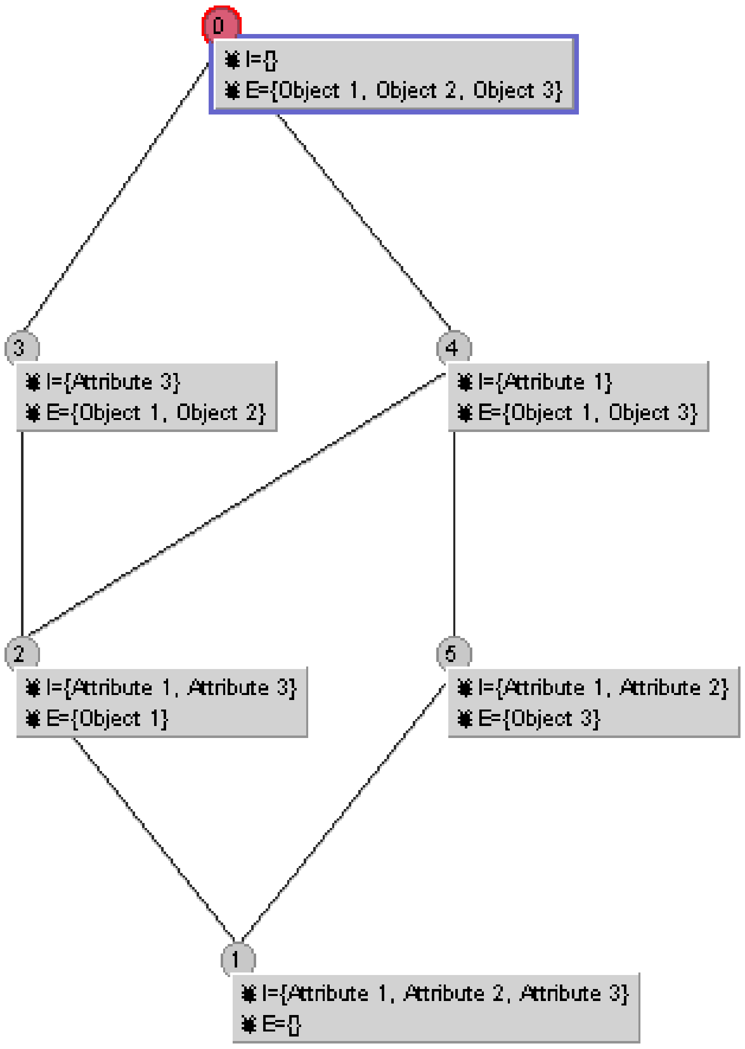

0
I={}
E={Object 1, Object 2, Object 3}
3
I={Attribute 3}
E={Object 1, Object 2}
4
I={Attribute 1}
E={Object 1, Object 3}
2
I={Attribute 1, Attribute 3}
E={Object 1}
5
I={Attribute 1, Attribute 2}
E={Object 3}
1
I={Attribute 1, Attribute 2, Attribute 3}
E={}

Figure 1.1: Concept lattice of formal context provided in 1.1

well. (Where an extension is a subset of the examples, the intention is the description and relations are a generalization.)

Galois sub-hierarchy (GSH) - the Galois sub-hierarchy (GSH) of a concept lattice is the partially ordered set of labelled elements only [5]. The example of GSH generated based on Table 1.1 is displayed in Figure 1.2 (lattice is generated using Maarja's application [17]). It is worth being noticed, that lattices displayed in Figures 1.2 and 1.1 are generated based on the same context, but the Galois lattice is easier to read and understand than the initial context lattice.

GSH was first introduced in 1993 by Godin et al. [5] and was a unit of study of different researches since then. Its advantage is in a simplified representation of original concept lattices, which leads to a better perception of the context due to a

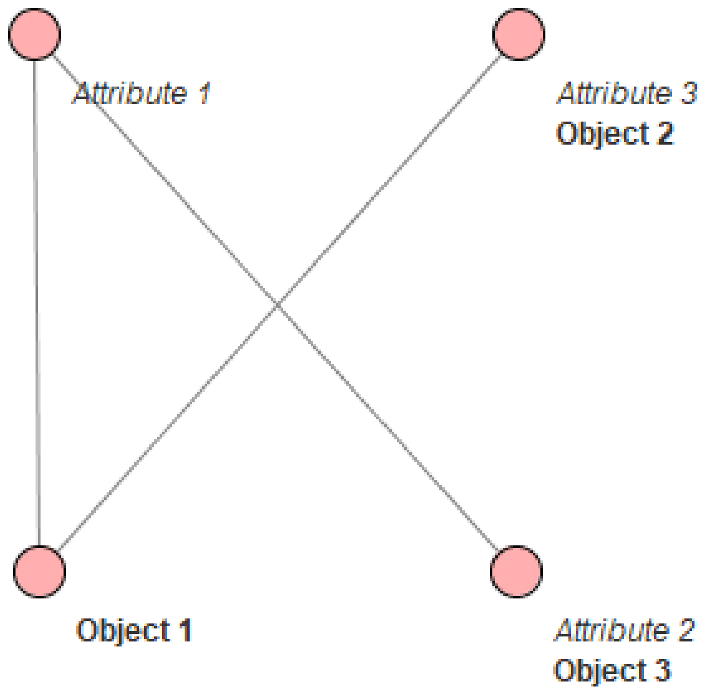


Figure 1.2: Galois lattice of formal context provided in 1.1

restricted number of elements being visualized.

1.1.2 Existing soulutions

At the moment there is no solution that is capable of visualizing GSH for multi-layered systems at various abstraction levels. But there are some tools, which were examined and taken into account during the research, for GSH generation based on binary formal context. In the scope of this work only tools starting from the year 2005 and later are taken into consideration, however, the author is aware of the fact that there is also a research paper that provides an overview and a brief comparison of older FCA related tools [21].

A brief overview of each of the considered tools is provided below. Test output for each tool was generated using the same test context input, which can be found in Appendix 1.

Galicia

The most known alternative tool for visualizing conceptual lattices and GSH graphs is Galicia [20]. It focuses on FCA tools in general, not only on GSH. Thus it has a lot of functionality that is out of the scope of the thesis.

Galicia provides an implementation of different algorithms for building GSH, but the visual output is hardly readable (the readability decreases drastically with increasing complexity of the context,i.e. a number of objects, attributes and relations) and there is no support of CSV input files (which is a big inconvenience, as CSV file format is widely used for describing formal context). Also, the last update of the program was made in 2005, which means that it was not changed for about 14 years, which makes it outdated.

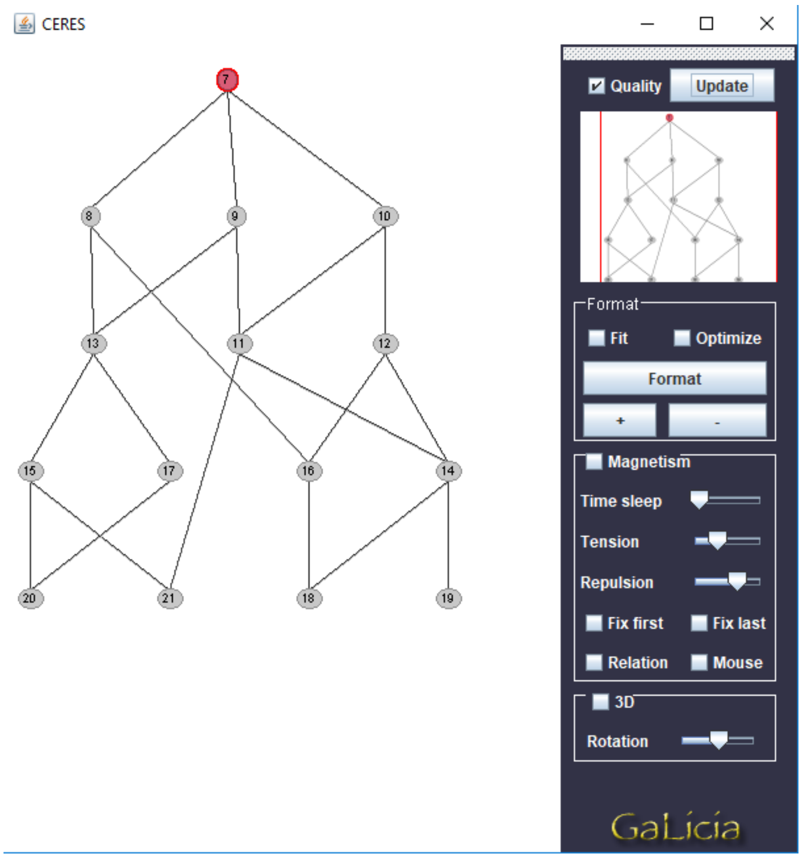


Figure 1.3: GSH generated by Galicia

GSH visualization tool - Maarja Raud

Visualization tool developed by the student of TUT - Maarja Raud [17] allows to generate GSH graph based on a binary matrix which is provided in CSV format. It allows to export generated diagram in JPG format, change the position of the elements on the picture. It also shows connected elements of the concept by highlighting the edges and selected node itself. Unfortunately, the collisions are not handled properly, alignment of elements makes it difficult to read and understand a hierarchy where there is a lot of concepts. The export functionality is absent and it is not possible to change the context data on the run: in order to do that, a new CSV should be imported.

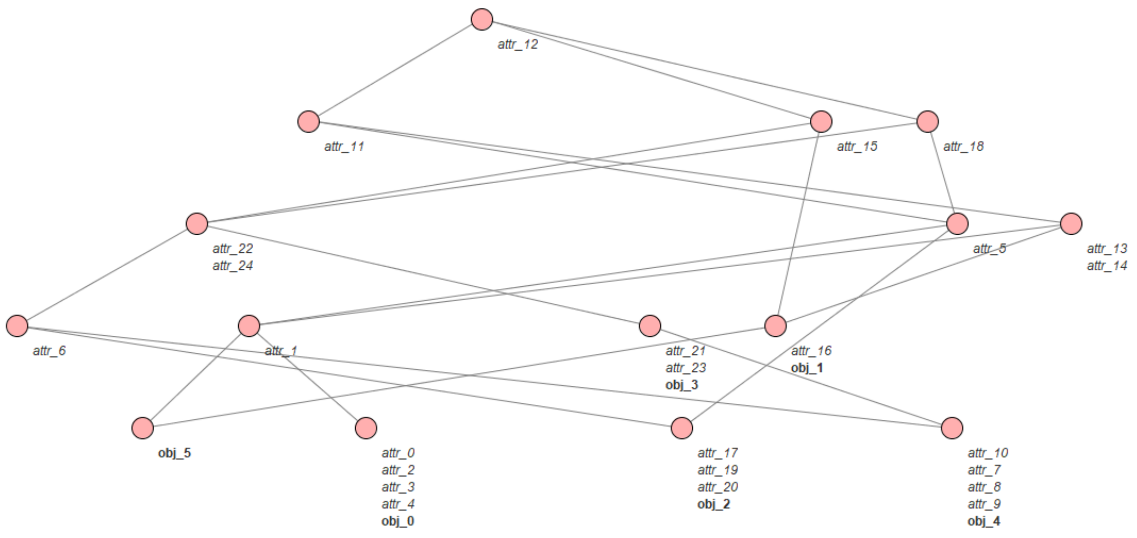


Figure 1.4: GSH generated by Maarja's visualization tool

GSH visualization tool - Kristo Aun

Visualization tool developed by the student of TUT - Kristo Aun [17] allows generating GSH graph based on a binary matrix which is provided in XML format. It is also possible to change elements of formal concept in the program as well as generate random context based on given parameters. Image export is not supported, but it is possible to export DOT file. The generated graph is also adjustable: it is possible to change the size, name, and position of the elements. Unfortunately, CSV format is not supported.

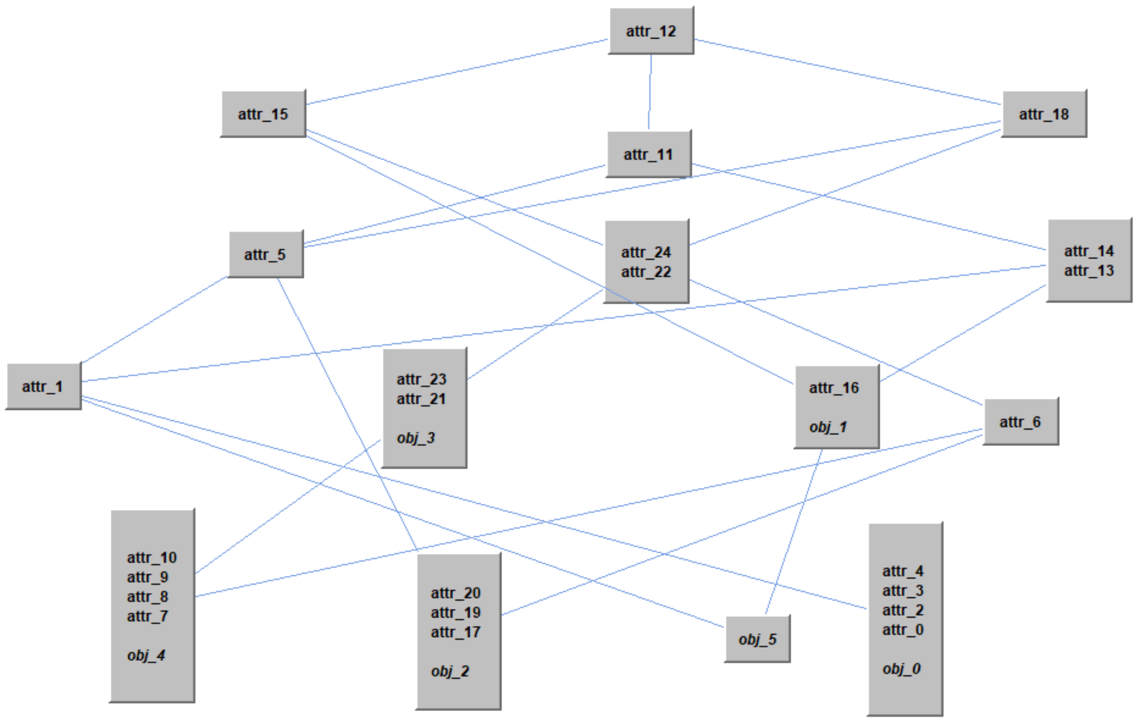


Figure 1.5: GSH generated by Kristo's visualization tool (position of the elements is adjusted in order to save space)

1.2 Research questions

The complexity of the stated goal is related to the nature of the tool, as it should be dynamic and flexible with regard to system description. It should be possible to describe and change system parts on the run and all layers and components should be aware of conducted changes, meaning the information should be updated automatically in real time. At the same time, the possible complexity of the system under description should be taken into account: the system may have 1...n layers, where each layer may have some number of elements (objects and attributes from point of view of formal context) and each set of elements may be divided into subsystems. Each subsystem in its turn may have a subsystem of itself and so on, meaning the depth of such hierarchy is not restricted.

During the research, a method of handling multiple layers, their sub-components and related abstractions was developed. It was important to create an appropriate data structure along with finding the proper way of visualization of complex data, in order to reduce or fully eliminate (where possible) the need of adjusting the data manually (which is a problem in case of existing solutions) and provide sufficient readability of the output to the end user.

Below is provided an overview of the main goals of the research and related research questions, which were considered during the work. Found solutions are described later in the document (corresponding section reference is provided for each research question separately).

- Definition of multiple layers of the system along with different abstraction levels [22]
  - How to handle multiple layers and their abstractions, considering that it should be possible to define sub-groups of elements - sub-systems - for each layer and relations between sub-systems should be generated automatically? See Sections 2.1, 2.2 for more details.
- Complex data handling
  - How to process complex data sets effectively? See Sections 2.1, 2.2 for more details.

- Lattice visualization
  - How to make displayable graph readable, avoiding collisions of the nodes as well as chaotic positioning of the elements? Existing solutions have known readability flaws [17], which make the process of working with visualization more complex. See Sections 2.2 and 3.1.1 for more details on technical solution and comparison with existing tools correspondingly.
- The design and validation of the tool
  - Design decisions in general (and technical decisions in particular), eg., selected technology or software design, are important and will have a direct influence on the outcome. Different possible solutions were examined during this research and appropriate choices were made (see Section 2.1 for more details on application design and Section 3.1 for validation details).
- Utility of the solution
  - How to make sure that such a tool is useful for case studies? Case study based validation and interpretation of the results is provided in Section 3.2.

1.3 Methodology

To solve the stated research goals and to answer stated research questions (provided in Section 1.2), a careful study was conducted. The result of the research is the solution, which is designed based on the acquired data, conclusions and findings obtained during the work. Previous related experience (researches, existing solutions, etc.) was taken into account and laid a foundation for the derived solution.

The design science (DS) research method was used by applying the design science research methodology (DSRM) [23] with some insignificant adjustments (caused by the nature of the work and topic). The DSRM was chosen primarily due to the following aspects of the work that satisfy DS main principles:

- The produced outcome of the research is an artifact, which is created to address the stated problem.
- The stated problem is unresolved at the moment.
- And, although there is no specific business which would make demands on the solution, the outcome of the research will provide potential interest for analysis phases in a lot of different fields [12] and thereby it is considered to be an important business problem.
- The entry point for the research is also in conformity with DSRM.

Furthermore, the DSRM activities are in conformity with the type of the stated problem and correspond to the planned process of validation and interpretation of the results. List of activities (i.e., research steps) along with short definitions is provided below.

1. Problem identification and motivation
   At this step, the research problem was defined and relevant knowledge base studied. As an outcome of this step required theoretical background was obtained and the possible valuable solution was proposed (see Chapter 1).

2. Define the objectives of a solution
   This stage includes the definition of general objectives. Requirements were

set and specific criteria determined. This step resulted also in knowledge of existing technologies and their possible application in the scope of the solution (see Section 1.1.2 for an overview of existing solutions and Section 2.1 for requirements and overview of technologies).

3. Design and development
   At this step, an artifact that solves the research problem was designed and developed (see Section 2.1).

4. Demonstration
   This step is needed to prove that created artifact really works and to show how. The demonstration was conducted on a test case and results are provided in Section 3.2.

5. Evaluation
   Here evaluation of the demonstration results was conducted by comparing the objectives with results acquired in the previous step. Acquired results were also compared to the existing solutions (comparison is described in Section 3.1.1 in details). The evaluation is covered in details in Section 3.3.

6. Communication
   This thesis itself is a way of communicating the problem, solution, and utility to relevant audiences. To ensure that interested parties can take maximal benefit of the proposed solution, it was also made open source (see Section 2.2 for repository link), thus anyone can participate in the future improvements (list of future improvements is provided in Section 3.3.1) and/or download and use the solution without restrictions.

Chapter 2

GSH visualization tool

2.1 Application design

The designed tool is a web application as today web technologies are capable enough to perform tasks that are required to achieve goals stated in the thesis (which was also confirmed during conducted research of available technologies). Furthermore, in this particular case web application is considered to be more convenient than standalone application, as it does not require installation from the end user and is platform independent, while standalone application may require installation and may not work on all systems (for example, desktop application will not work on mobile phone, or some incompatibilities may occur between different OS versions).

2.1.1 Requirements

The requirements designed based on the objectives of the work are provided below. Each objective (upper item in the list) has its own set of requirements (sub-items of the list). Requirements documented as user stories, due to the nature of the system under development. If there are any additional (specifying) requirements, then they are provided below a user story that they belong.

In the given context, "user" means a person, who performs FCA on a specific system. The initial motivation of the user (that may stand behind the need to conduct the FCA) and/or his/her field of interest (i.e., research field, job position, company, etc.) are out of the scope of this thesis and are not taken into account.

1. **Definition of multiple layers of the system along with different abstraction levels**

   1.1. As a user, I want to be able to load CSV definition of a context, so that I do not have to manually describe the context contained in the existing files.

   1.2. As a user, I want to be able to export CSV definitions of defined contexts, so that I can process them outside of the application.

   1.3. As a user, I want to be able to define relations between different system layers, so that I can perform an FCA on a multi-layered system.
   - Relation between layers must be specified by defining relations between the contexts.
   - Contexts can be related only if attributes (columns) of upper context are the objects (rows) of the lower context at the same time.
   - The fulfillment of condition defined above must be automatically controlled by the system.

   1.4. As a user, I want to be able to modify defined context, so that I can adjust it as needed during the analysis.
   - Must be possible to apply modifications on the graph directly.
   - All elements of the system that are related to modifications must be aware of applied changes.

   1.5. As a user, I want to be able to define subsystems for each layer, so that I can describe a hierarchy of the system under analysis.
   - Number of the subsystems for each layer must not be restricted.
   - It must be possible to define sub-subsystems for other subsystems.
   - Number of the subsystems for each upper subsystem must not be restricted.
   - Depth of the subsystems' hierarchy must not be restricted.

2. **GSH lattice visualization**

   2.1. As a user, I want to be able to see the graphical representations between the relations of the concepts, so that I can analyze them efficiently.

  - Collisions between elements and frames should be decreased or fully eliminated (where possible).
  - Concepts must be represented as nodes.
  - Relations must be represented as edges.
  - Attributes must be positioned at the top of the node label.
  - Objects must be positioned at the bottom of the node label.
  - Relations maximum of two layers can be defined at a time (i.e., one context).

2.2. As a user, I want to be able to change the position of the elements on the graph, so that I can adjust the visual output as needed.

2.3. As a user, I want to be able to export a picture of the graph, so that I can process it outside of the application.

2.4. As a user, I want to be able to see relations between different abstraction levels of the context, so that I can have a better picture of the system architecture.

  - Abstraction level change should be available only if at least one lower level (subsystem) of the context is defined.

2.1.2 Technologies

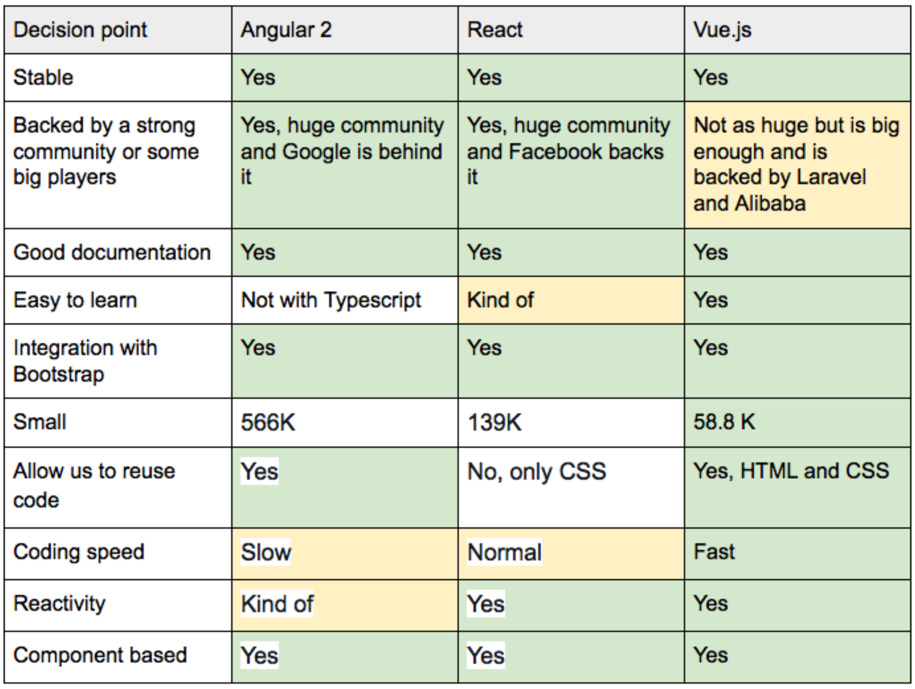
This section contains an overview of the main technologies that were studied and used during the research in order to achieve the objectives of the thesis. Some of the technologies are based on scientific papers, thus providing a solid foundation for the proposed solution.

Back-end - Spring Boot + Java

Spring Boot is an extension of Spring framework [24], which makes a configuration of the application as simple as possible. It allows to build a deployable artifact or to make an executable application, which embeds the server and can be started with two clicks from the local machine. It is very convenient as it allows to use the web application locally even if it is not available on some public server. Due to the fact that the developed tool uses at some extent existing Java project (for constructing GSH), Java is selected to be a back-end programming language of the solution.

Front-end - Vue.js

The study was conducted to understand which platform to choose for front-end development. There is an article [1] providing overview of most popular front-end frameworks, which is summarized in table as follows:



Decision point	Angular 2	React	Vue.js
Stable	Yes	Yes	Yes
Backed by a strong community or some big players	Yes, huge community and Google is behind it	Yes, huge community and Facebook backs it	Not as huge but is big enough and is backed by Laravel and Alibaba
Good documentation	Yes	Yes	Yes
Easy to learn	Not with Typescript	Kind of	Yes
Integration with Bootstrap	Yes	Yes	Yes
Small	566K	139K	58.8 K
Allow us to reuse code	Yes	No, only CSS	Yes, HTML and CSS
Coding speed	Slow	Normal	Fast
Reactivity	Kind of	Yes	Yes
Component based	Yes	Yes	Yes

Figure 2.1: Summarized comparison of popular front-end frameworks provided in [1]

Taking into accordance advantages of the Vue.js [25], it was decided to use it as the main framework for front-end development.

GSH visualization - Cytoscape.js

The visualization of a GSH (graph, in fact) was one of the main problems of the work. In the course of the research different possible solutions were considered. There are multiple different studies conducted on visualization of hierarchical data, which are of interest from a number of perspectives [26] [27] [28], as well as graph visualization in particular [29] [30] [31].

Today there is an amount of open source web libraries, which provide graph generation functionality. For example, NGD: Angular Dependencies Graph [32]. However, after the closer examination, it became clear, that the library does not satisfy stated requirements and may need further changes of the source code in order to be capable to fulfill the objectives. Also, the selection of this library means the selection of the Angular platform [33] (which was not acceptable, see Subsection 2.1.2 for more details).

As an alternative to NGD, the Cytoscape.js [34] was considered. Cytoscape.js is presented as graph theory (network) library for visualization and analysis, it is based on scientific research, very dynamic and flexible, has solid documentation and big community. Due to the provided points, Cytoscape.js was selected as a graph visualization library in the scope of this work.

Other front-end technologies

The tool is built with a lot of different packages of different purposes. All of these packages can be downloaded from npm - JavaScript package manager [35]. Thus other technologies referred in this section are not referenced separately but provided in the list below by package name, which is an npm reference, meaning that each package can be found in an npm registry (online database of packages) by name if needed.

- **tabulator-tables**
  A Tabulator is a lightweight JavaScript table generation library. Used for handling context tables in the application.

- **papaparse**
  Papa Parse is a powerful in-browser CSV parser, which is used for parsing contexts uploaded as CSV files and vice versa: to create CSV data from context tables, defined directly in the application.

- **cytoscape-node-html-label**
  An extension for Cytoscape.js, which allows to generate custom node labels using HTML.

- **cytoscape-dagre**

  This JavaScript library allows to create a layout for directed graphs and used for hierarchical positioning of the elements on the canvas.

- **webpack**

  A module bundler for JavaScript. Used to handle JavaScript dependencies in the project: it creates dependency graph for modules and maps them accordingly.

  - **vue-style-loader** - a webpack module, that handles style loading for Vue.js application.
  - **vue-template-loader** - a webpack module, that handles template loading for Vue.js application.

- **axios**

  Axios is a promise based HTTP client. Used for sending HTTP requests to the REST application.

- **jquery**

  A JavaScript library that handles JavaScript and HTML interaction.

- **bootstrap-vue**

  For integrating Bootstrap 4 [36] components with Vue.js.

- **html2canvas**

  A library that allows saving pictures (screenshots) of HTML parts. It is used to save graph pictures instead of built-in Cytoscape.js functionality because Cytoscape.js native solution cannot handle custom labels, which are created using a cytoscape-node-html-label extension.

Other back-end technologies

- **Maven**

  Maven [37] is used as a build tool for the entire project. Among others, it handles proper npm execution during the build and also is responsible for including Maarja's JAR file into the project. Besides, it handles other back-end dependencies as well.

- **Project Lombok**

  Lombok [38] Java annotation-based library helps to avoid code repetitions and keep it clean and simple.

2.1.3 Architecture

The project consists of two main layers: back-end and front-end. Communication between layers takes place using HTTP requests: front-end sends structured formal context to the back-end REST service. After that context is being converted and processed in Java: GSH data structure is being generated and returned to the front-end, which handles a visualization of received results. The general architecture of artifact composition can be seen in Figure 2.2 and the communication between system layers is depicted in Figure 2.3.

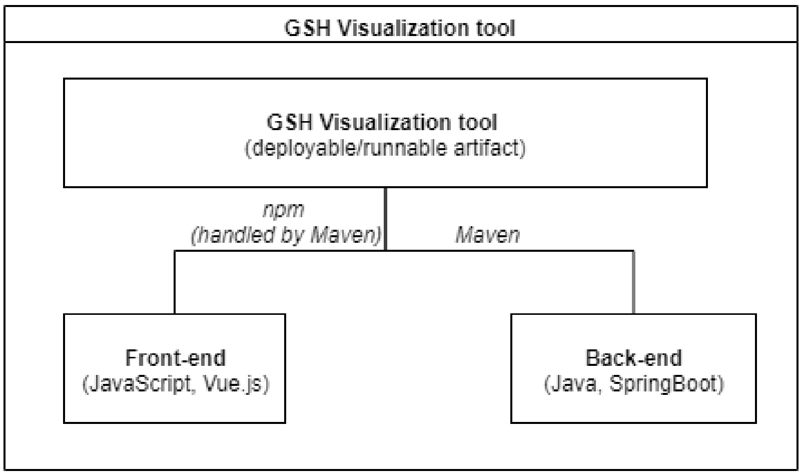


Figure 2.2: GSH visualization tool - architecture of artifact composition

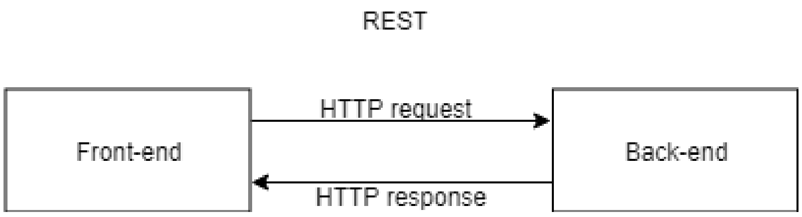


Figure 2.3: GSH visualization tool - communication between front-end and back-end

Back-end GSH generation logic is built on top of Maarja's visualization tool: existing solution is adjusted in accordance with new REST DTO's. The process of generating GSH consist of four steps:

1. Data parsing: objects and attributes of the CSV table are converted into the Relation object.
2. Concept lattice is constructed from parsed relations.
3. GSH lattice is built from the generated concept lattice.
4. GSH graph structure is simplified: object references are replaced with ID-s, custom hash function applied.

Then the graph structure is being returned via HTTP response to the caller. Back-end components and relations between them are depicted in Figure 2.4.

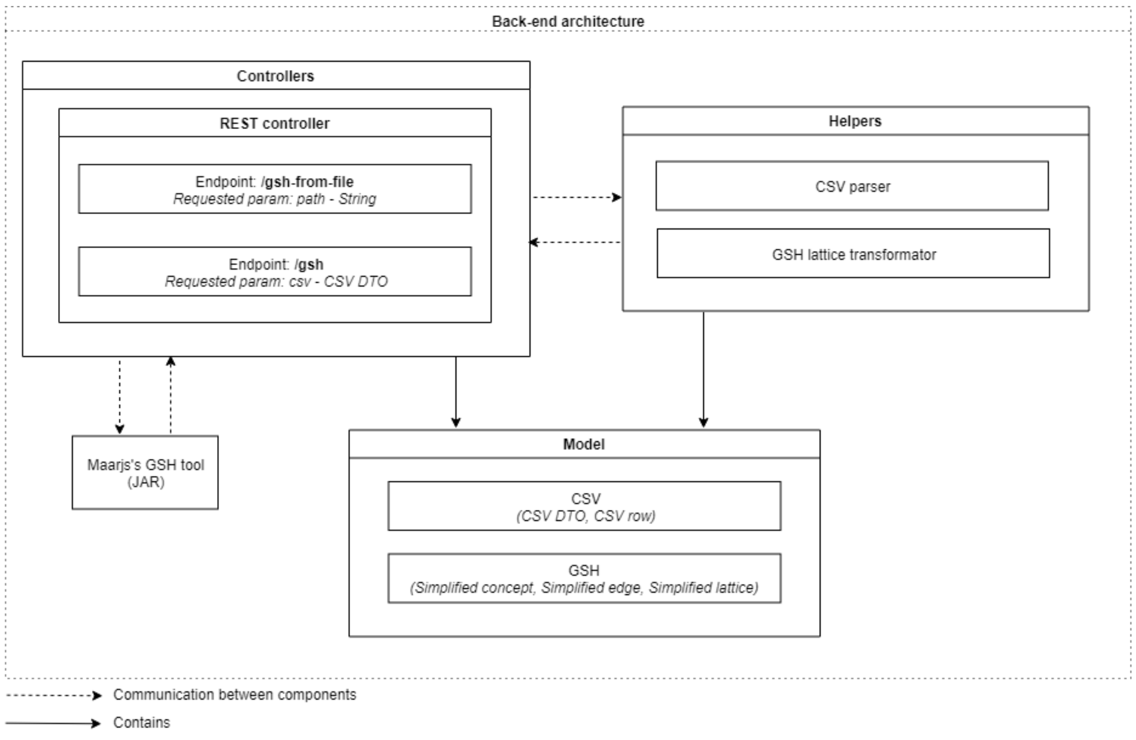

Back-end architecture
Controllers
REST controller
Endpoint: /gsh-from-file
Requested param: path - String
Endpoint: /gsh
Requested param: csv - CSV DTO
Helpers
CSV parser
GSH lattice transformator
Maarjs's GSH tool (JAR)
Model
CSV (CSV DTO, CSV row)
GSH (Simplified concept, Simplified edge, Simplified lattice)
Communication between components
Contains

Figure 2.4: Back-end components and their relations

Front-end part is built using Vue.js. It is divided into components and services, where components are main "building blocks" of the application and services provide general functionality (like HTTP communication, GSH visualization using Cytoscape.js, CSV parsing, etc.). See Figure 2.5 for more details. Vue components follow a single file pattern: that means that template, script and style parts are defined together in one file. It helps to organize the code and keep the structure clean. That does not mean that all code should be mixed together: each part of a single file can include a reference to the external file if needed. This approach is convenient when working with large and complex components. In the tool, such a component is "Data settings", which is responsible for handling and displaying all contexts of the system, defined by the user.

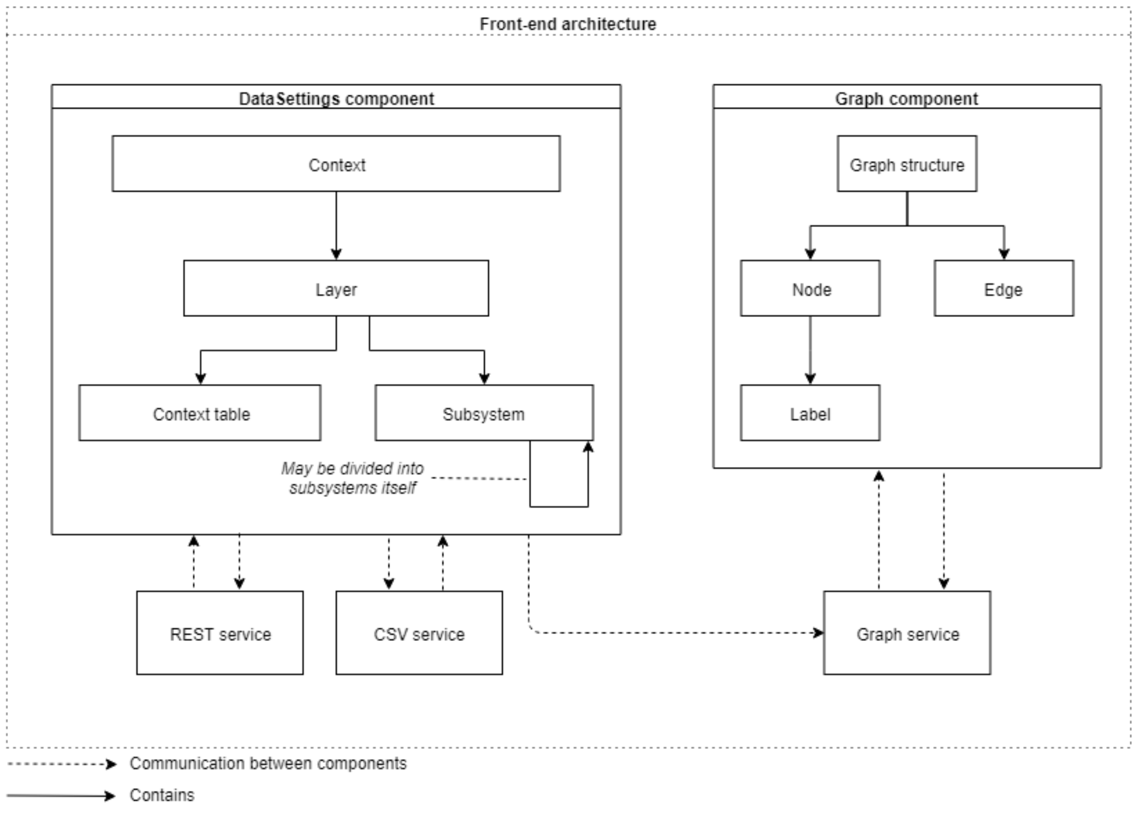


Figure 2.5: Front-end components and their relations

2.2 Implementation

The tool was implemented in correspondence with provided requirements and following developed design. The outcome of the development process is a standalone application, which can be built as an executable JAR file as well as a deployable WAR file, thus making it very flexible in terms of environmental requirements.

The source code is made public and is located at Github:

https://github.com/androlga/gsh-visualization-tool

The purpose of this decision is to allow anyone to use it and/or contribute to future development. Also, it provides an opportunity to receive more feedback on the tool from the users and therefore be aware of possibly missed flaws (if any) and unexpected users' needs, which may lay a base for future works.

This section contains the main highlights of the implemented solution along with a description of occurred problems and how they were handled.

2.2.1 GSH transformation

GSH structure is a base for the graph visualization. Unfortunately, the existing implementation of GSH generation resulted in a quite complex structure, which contained a lot of nested objects for each node. Thus, the result was heavy and identification of elements was not trivial, as they contained no explicit identifier. It was not possible to use it on the front-end side, as its processing took a lot of time and failed completely for complex contexts with a lot of nested objects.

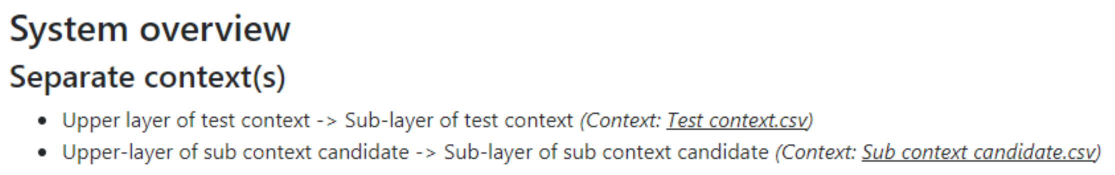
To solve that problem, the initial GSH structure is simplified using a custom transformer. During the transformation process, redundant nested objects are being eliminated and appropriate hash is being given to each object (edge and concept), so it could be easily identified.

2.2.2 Context layers

Each binary context initially has two layers, where objects (rows) are the elements of the upper row and attributes (columns) are elements of the lower row. To define a new system layer, new context with appropriate relation should be defined. In this case, objects of the lower context should be the same as the attributes of the upper context (as it is the same layer). Therefore, for two layers - 1 context, for three layers - 2 contexts and so on.

All information about existing contexts and related layers are stored in a JavaScript object, which provides access to values by keys. So each defined context has its own unique key in the system. Each context has in its turn a reference to sub-context (if any), so the application could find related objects.

A list of sub-context candidates, as well as information about context relations, is constantly updated if related data change is detected by Vue. An example of the system state with two separate contexts is provided in Figure 2.6. In this case, it is possible to create a relation between the contexts, which can be seen from Figure 2.7.

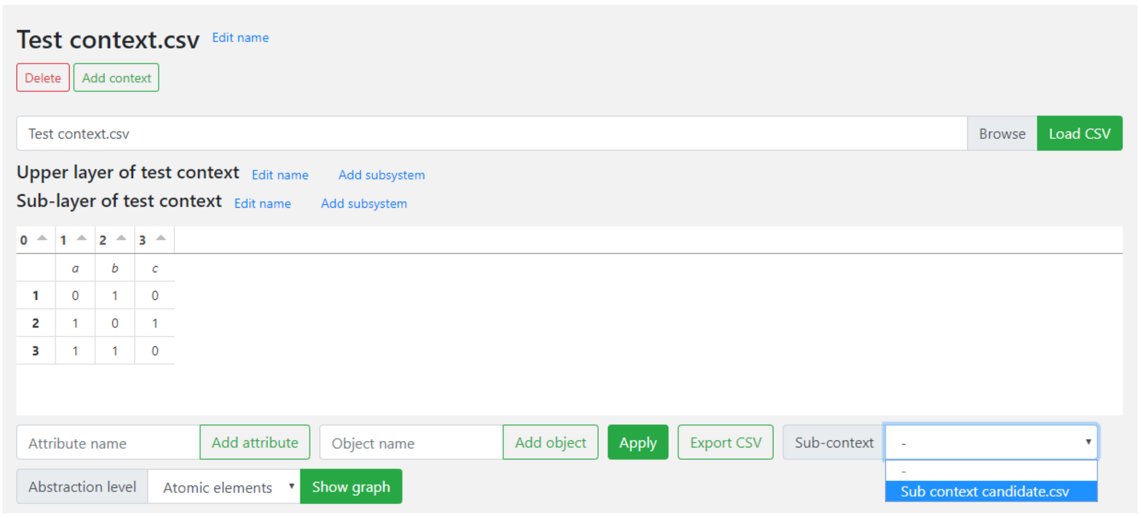


System overview

Separate context(s)

- Upper layer of test context -> Sub-layer of test context *(Context: Test context.csv)*
- Upper-layer of sub context candidate -> Sub-layer of sub context candidate *(Context: Sub context candidate.csv)*

Figure 2.6: Screenshot from the application - overview of system with two separate contexts



Test context.csv Edit name

Delete Add context

Test context.csv Browse Load CSV

Upper layer of test context Edit name Add subsystem

Sub-layer of test context Edit name Add subsystem

0	1	2	3
	a	b	c
1	0	1	0
2	1	0	1
3	1	1	0

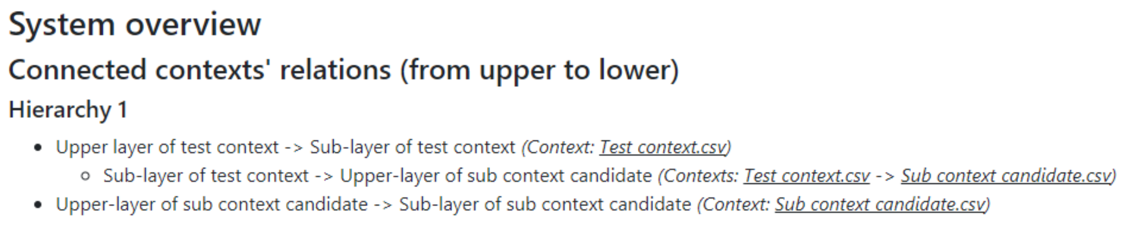
Attribute name Add attribute Object name Add object Apply Export CSV Sub-context -

Abstraction level Atomic elements Show graph

-
Sub context candidate.csv

Figure 2.7: Screenshot from the application - system provides choice of sub-context

If the context is assigned, then the system data changes immediately. Updated system overview is depicted on Figure 2.8.



System overview

Connected contexts' relations (from upper to lower)

Hierarchy 1

- Upper layer of test context -> Sub-layer of test context *(Context: Test context.csv)*
  - Sub-layer of test context -> Upper-layer of sub context candidate *(Contexts: Test context.csv -> Sub context candidate.csv)*
- Upper-layer of sub context candidate -> Sub-layer of sub context candidate *(Context: Sub context candidate.csv)*

Figure 2.8: Screenshot from the application - overview of system with two related contexts

2.2.3 Hierarchical subsystems

Each context contains two layers, each layer may contain from 0 to n sub-hierarchies. Each sub-hierarchy, in turn, may contain lower sub-hierarchies with no restrictions on hierarchy depth. Thus, the relations between sub-systems can be represented as a tree. Defining data structure, in this case, was a little challenge, as storing the nested data decreases performance and sometimes even may cause call stack excess related problems. Thus it was decided to store all subsystems of the specific context as divided to upper and lower (with regard to upper and lower layers of the context, i.e. objects and attributes). Each layer has its own set of subsystems, where each subsystem only contains a value of hierarchy level. Also, there is a data set for each layer (i.e., max 2 data sets per context), that holds all relations of hierarchies as key-value pair, where the key is an id of lower sub-system and value - id of parent sub-system. This approach allows to save performance on data updates and has been proven to be suitable for defining sub-systems as well as finding requested abstraction levels for selected context. An example of how multiple layers of sub-systems may look like is provided in Appendix 2. Appendix 3 contains an illustration of sub-systems hierarchy with no lower levels.

2.2.4 GSH visualization

For GSH visualization Cytoscape.js is used along with additional extensions. Hierarchical layout is provided using a cytoscape-dagre library and node labels are described (customized) and created using a cytoscape-node-html-label extension.

There is a separate container for graph layout on the page, to which Cytoscape instance is assigned. Each time when GSH structure is received from the REST service, the data is being sent to GSH service and graph is being drawn.

The graph provided in Figure 2.9 demonstrates the visual output produced by the application. The graph based on the same context (see Appendix 1) that was used for demonstration purposes in section 1.1.2. Each node of the graph is labeled, relations between nodes are represented by edges and hierarchy is defined by arrows. All elements of the same hierarchy level are positioned in one line and there are no collisions of elements. It is possible to zoom in/zoom out the layout, to move the graph or it's elements, to change the layout (from Hierarchical to Circle, Cose, Grid or Random) and to save the picture of the graph that is currently positioned on the layout. The circle layout is provided as an example in Figure 2.10.

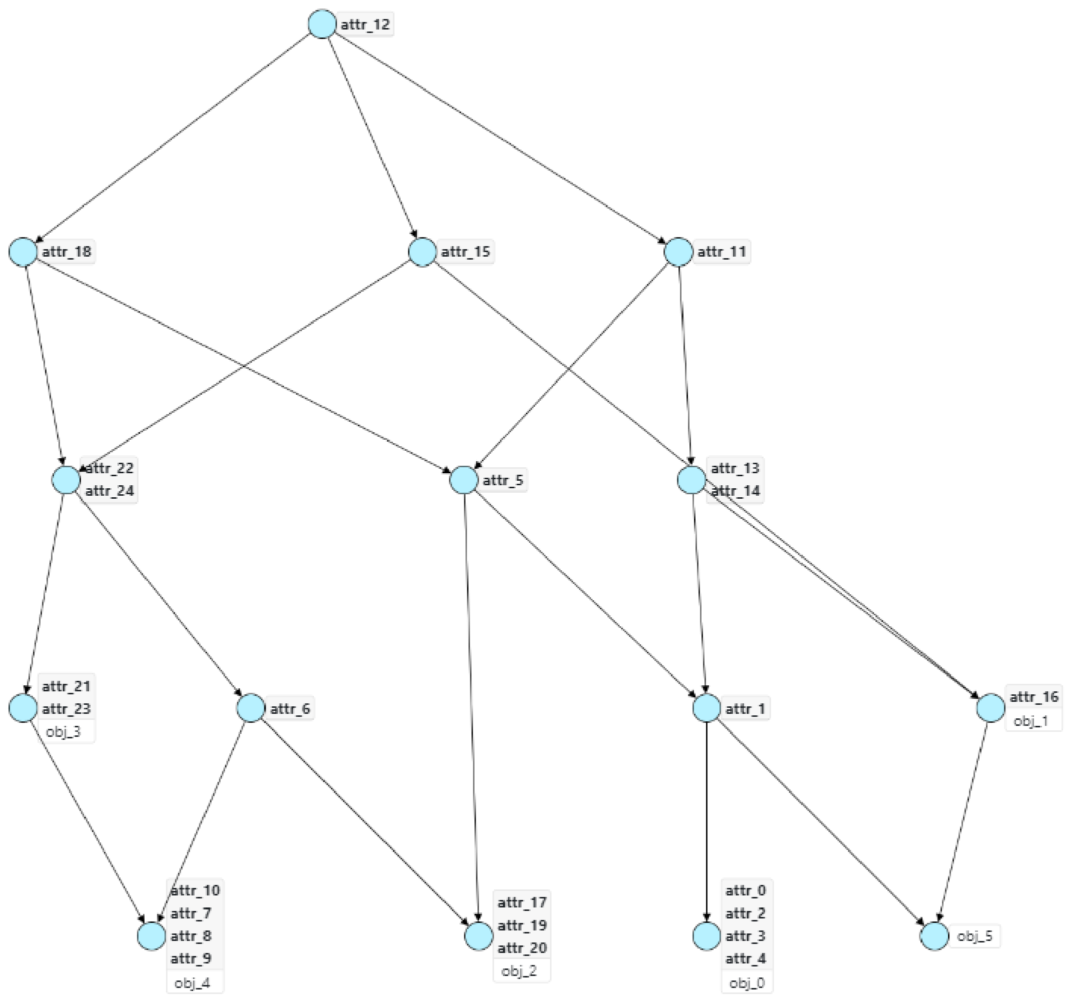


Figure 2.9: GSH visualization created by application - Hierarchical layout

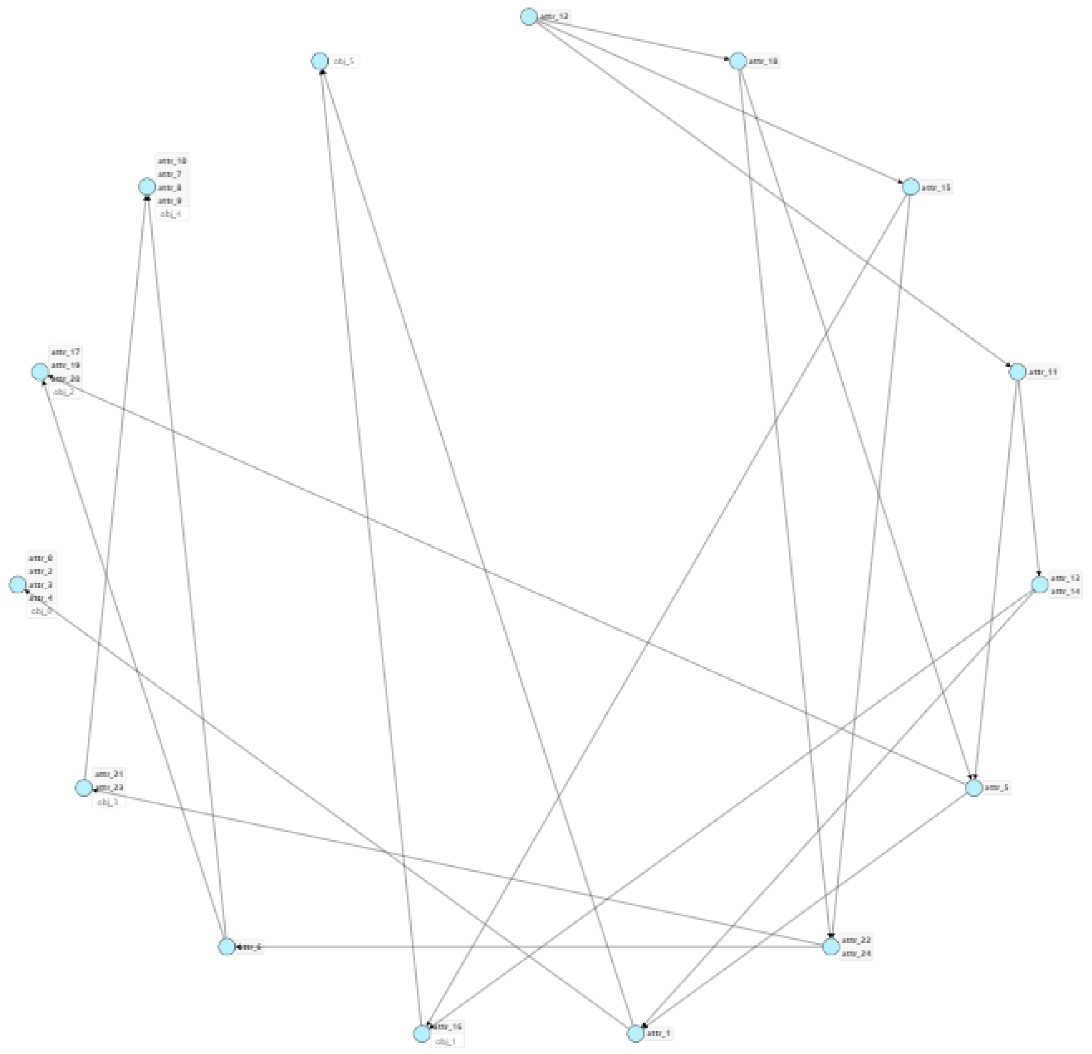


Figure 2.10: GSH visualization created by application - Circle layout

Chapter 3

Validation

3.1 Validation

The validation section describes actions that were applied to check the utility of the solution, to check if the solution works as expected and does it correspond to stated objectives or not. The results of the validation are evaluated and discussed at the end of this section.

3.1.1 Comparison with existing solutions

This section provides a comparison of the created tool with existing solutions (overview of which can be found in Section 1.1.2). As none of the solutions has the functionality of sub-systems and data visualization on different abstraction levels, related comparison points are not provided in the list. The functionality of defining contexts and visualizing GSH is the main subject of comparison.

As an entry point for the comparison, GSH generated by existing tools is used. GSH visualization performed by Galicia, Maarja's and Kristo's tools can be found in the Figures 1.3, 1.4 and 1.5 respectively. The visualization output provided by the tool developed in this thesis is depicted in Figure 2.9. All outputs are generated using the same context (see Appendix 1) and thus are based on the same data.

- **Galicia tool**

  The elements on the generated diagram are positioned correctly, collisions are eliminated and structure is easily readable, which is also true for the developed tool. Unfortunately, Galicia has an outdated interface: it requires multiple actions from the user in order to perform one step, which may be inconvenient. It also does not display node labels (objects and attributes) by default, which makes it harder to analyze and requires additional actions to fix that. For comparison, the developed tool displays labels in an easily-readable form and has an intuitively understandable responsive interface. Both solutions highlight the selected node and allow to adjust positions of the elements directly on the graph.

Galicia also has some features that are outside of the scope of this thesis and were not stated as requirements (for example, 3D representation of the graph or magnetism option applied to nodes), thus not considered during comparison as well.

- **Maarja's GSH visualization tool**
  This solution accepts CSV files as context definition, but unfortunately does not generate a context table or provide CSV export feature. The positioning of the elements is also a problem, as collisions are present and it is hard to read some of the labels. From the other hand, it has a nice feature of highlighting the selected concept and related edge, which is absent in the proposed solution (implemented tool highlights selected element only, not the references).

- **Kristo's GSH visualization tool**
  This tool allows editing a context table and import/export the data. Unfortunately, there is no CSV support, but the XML file format is supported. The elements positioned hierarchically, but some collisions still take place. There is also no zooming functionality provided for the diagram.

Summary table of different features comparison is provided below in Table 3.1. The features that absent in all tools (or vice versa) are not included. The table shows that the implemented solution contains the greatest number of features regarding GSH visualization and context definition. The comparison points were set based on the stated objectives of the research.

	Galicia	Maarja's tool	Kristo's tool	Developed tool
Handles collisions properly	yes	-	-	yes
Displays names of objects and attributes on generated graph (by default)	-	yes	yes	yes
Editable context table	yes	-	yes	yes
CSV import	-	yes	-	yes
CSV export	-	-	-	yes
Data export/import in other formats (except images)	yes	-	yes	-
Highlights selected element and it's edges at the same time	-	yes	-	-
Total	3	3	3	5

Table 3.1: Summary of comparison of implemented tool with existing solutions

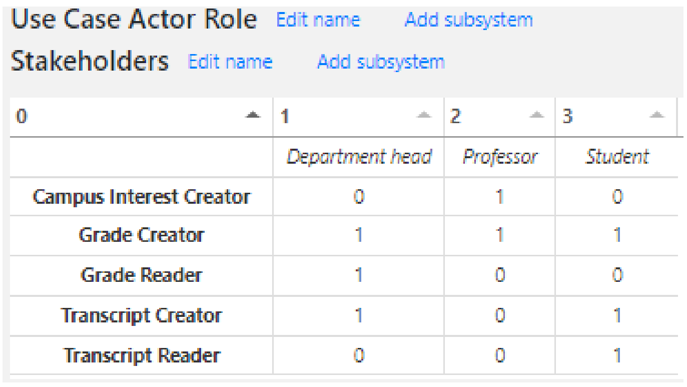
3.2 Case study

In order to validate the utility of the developed tool, the case study was considered. The article that handles the creation of the security model for student record system [2] is used as an entry point for this validation.

The student record system that is used in the article to illustrate the development of a security model contains four layers that are as follows (ordered from upper to lower):

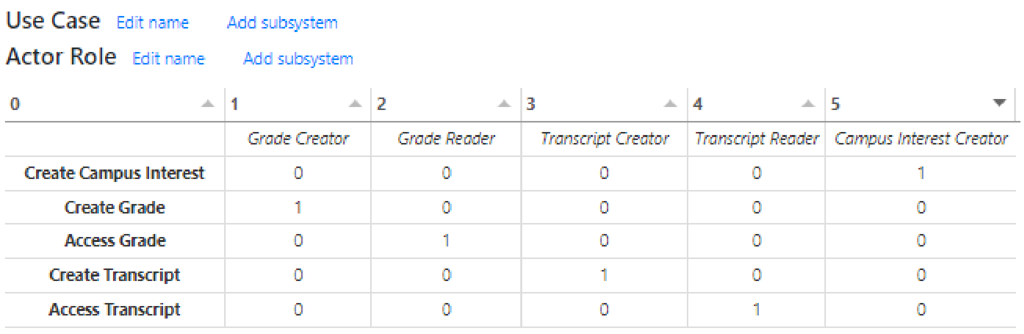
1. Stakeholders
2. Use Case Actors
3. Use Case
4. Data Entity (Class)

The binary matrices (contexts) that represent relation of the layers are defined as depicted in Figures 3.1, 3.2 and 3.3 below. The overview of the relations between contexts is depicted in Figure 3.4. The overview helps to understand if the system design is correct at the step of defining layers. Also if there is intended to be a relation between two layers, but the tool does not allow that, that means that there are some mistakes in the definition of the elements (some elements are missing or names of the elements do not match). (Which is not the case of the considered system.)



0	1	2	3
	Department head	Professor	Student
Campus Interest Creator	0	1	0
Grade Creator	1	1	1
Grade Reader	1	0	0
Transcript Creator	1	0	1
Transcript Reader	0	0	1

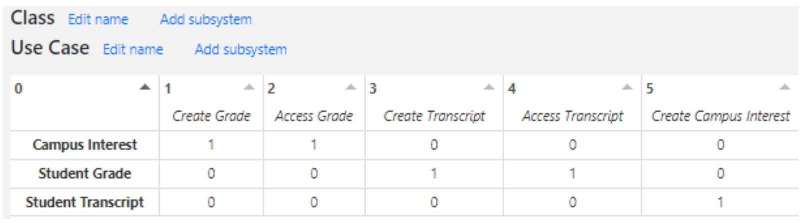
Figure 3.1: Screenshot - Stakeholders / Use Case Actors context



Use Case Edit name Add subsystem
Actor Role Edit name Add subsystem

0	1	2	3	4	5
	Grade Creator	Grade Reader	Transcript Creator	Transcript Reader	Campus Interest Creator
Create Campus Interest	0	0	0	0	1
Create Grade	1	0	0	0	0
Access Grade	0	1	0	0	0
Create Transcript	0	0	1	0	0
Access Transcript	0	0	0	1	0

Figure 3.2: Screenshot - Use Case Actors / Use Case context



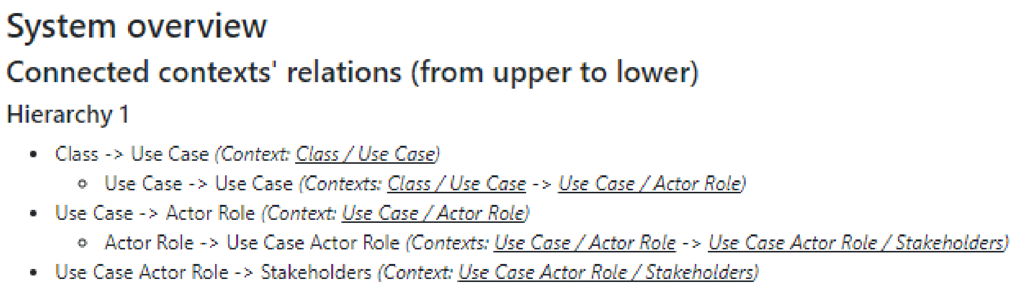
Class Edit name Add subsystem
Use Case Edit name Add subsystem

0	1	2	3	4	5
	Create Grade	Access Grade	Create Transcript	Access Transcript	Create Campus Interest
Campus Interest	1	1	0	0	0
Student Grade	0	0	1	1	0
Student Transcript	0	0	0	0	1

Figure 3.3: Screenshot - Use Case / Data Entity context

The Use Case diagram for the student record system is provided in Figure 3.5. The original diagram is used in order to illustrate how it differs from the generated GSH output. The Use Case illustrates relations of Use Case / Actor Role layers and at the same time shows the relations between Use Case Actors, based on the Stakeholders / Use Case Actors context (see Figure 3.1). The visual output generated by the GSH tool, from the other hand, allows to see these relations separately (Figures 3.6 and 3.7). It also provides a better overview of the relations, because of the clear hierarchical structure. Using generated GSH it is easy to see if any of the components were defined wrongly.

For example, lets change a Use Case / Data Entity context a little bit and lets say that *Student Transcript* Class is related to *Create Grade* Use Case. Correspond-



System overview

Connected contexts' relations (from upper to lower)

Hierarchy 1

- Class -> Use Case *(Context: Class / Use Case)*
  - Use Case -> Use Case *(Contexts: Class / Use Case -> Use Case / Actor Role)*
- Use Case -> Actor Role *(Context: Use Case / Actor Role)*
  - Actor Role -> Use Case Actor Role *(Contexts: Use Case / Actor Role -> Use Case Actor Role / Stakeholders)*
- Use Case Actor Role -> Stakeholders *(Context: Use Case Actor Role / Stakeholders)*

Figure 3.4: Screenshot - overview of the relations between the contexts

ing GSH structure was generated and is depicted in Figure 3.9. This data change illustrates how easy it is to understand that something in the model is wrong just by looking at the corresponding GSH diagram.

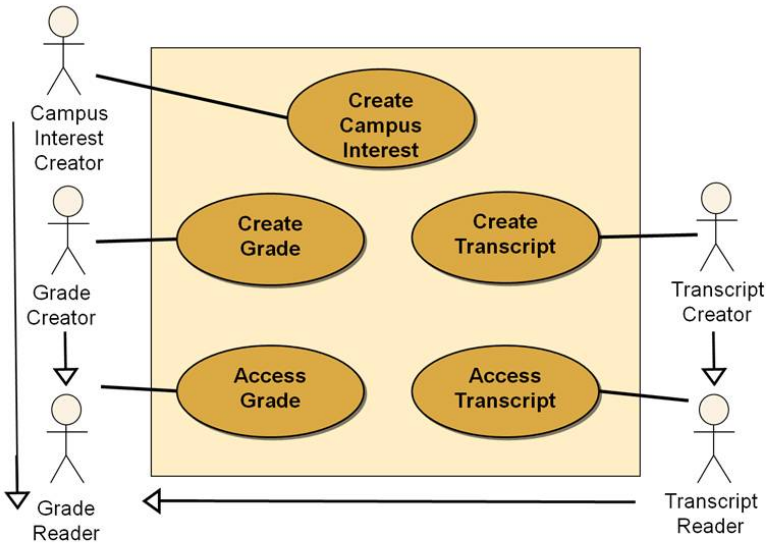


Figure 3.5: Use Case Diagram for Student Record System [2]

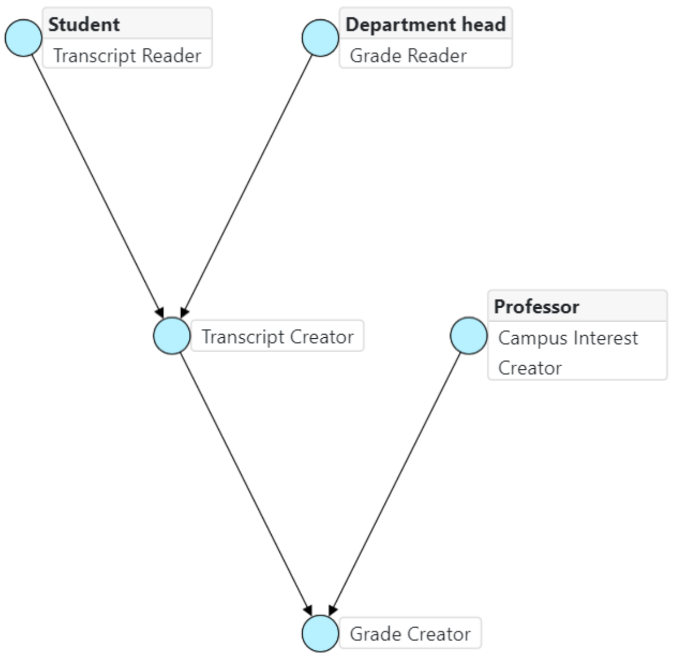


Figure 3.6: GSH generated based on Stakeholders / Use Case Actors context

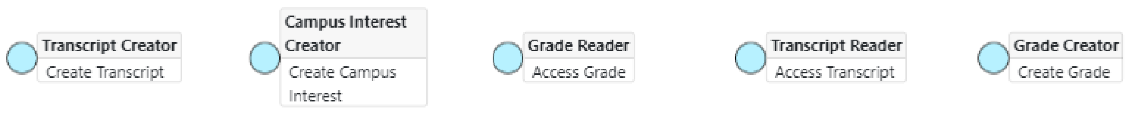


Figure 3.7: GSH generated based on Use Case Actors / Use Case context

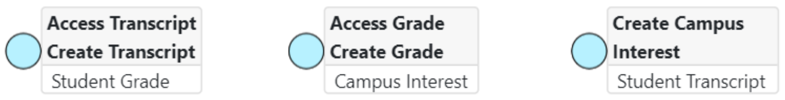


Figure 3.8: GSH generated based on Use Case / Data Entity context

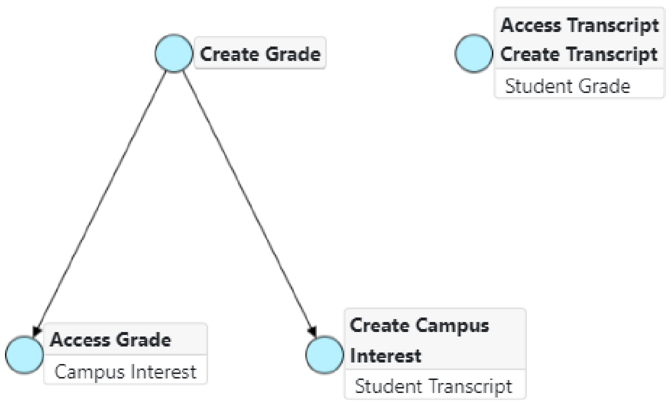


Figure 3.9: GSH generated based on changed Use Case / Data Entity context (which contains consciously made analytic error in it)

3.3 Evaluation and discussion

During the validation process, the comparison of the implemented GSH visualization tool with existing solutions was conducted. Also, the utility of the tool was demonstrated on the case study, which shows the implementation of the tool on the real use case. This section contains a description of the interpretation of the validation results and conclusions based on the obtained outcome.

A demonstration conducted on a case study illustrates the utility of the developed solution applicable to a real use case. First of all, the novel (comparatively to existing solutions) functionality (as support of relations between multiple layers, different abstraction levels and hierarchical definition of sub-systems) provides a possibility to define complex systems and conduct FCA on them, using structured and easily readable visual output. The provided use case also shows, how errors in the system design can be easily spotted thanks to the implemented tool. Secondly, all relations are easily modifiable and can be exported in order to be processed outside of the system. Thirdly, the solution of GSH visualization itself, although, is not unique, but is competitive, as it is capable of solving main related problems which may remain unresolved in other existing solutions (e.g., positioning of the elements and collisions handling).

Comparison of the GSH visualization functionality with existing tools also showed that the tool solves the greatest number of the stated problems. In particular, objects and attributes are easily distinguished, elements (nodes) are positioned hierarchically and the positioning is correct (no collisions, elements of same hierarchy level are positioned on the same line horizontally, etc.). GSH visualization tool has fewer functions in comparison with Galicia (e.g., 3D layout), but simpler interface. Also, these additional functions do not influence on the solution of the stated problem directly, thus are not taken into accordance.

Thereby, based on the provided evaluation may state, that results obtained as an outcome of the validation process are positive. The results show, that the implemented tool is working as expected and satisfies stated requirements.

3.3.1 Possible future works

There is a number of improvements and future developments that are out of the scope of the thesis but could be implemented in the future. The source code structure allows to easily make modifications or extensions of existing objects. Also, the code is located in the public repository so anyone can contribute. Thus, the conditions for the implementation of the improvements and/or additional functionality are favorable. The list of the proposed future works is as follows:

- **Support of different import/export standards.** Appropriate research should be conducted in advance in order to get a knowledge of which standards (besides CSV) could be useful for users.
- **Support of other GSH generation algorithms.** At the moment the native algorithm from the Maarja's tool is used (developed by Ants Torim). There is a study that provides a comparison of different algorithms for building Galois sub-hierarchies [39] as well as later studies on algorithmic solutions of the problem [40], which also may be taken into account.
- **UI design.** At the moment the design is responsive and simple. But the UX (User Experience) part may not be optimal. Additional research required in order to state the objectives on the improvement of the UX. For example, maybe more complex design (e.g., theme) could help to organize the space better.
- **New graph features.** Which includes (but not limited to), for example, adding editable labels and highlighting the hierarchy of the selected node.
- **Possibility to always displaying one graph per context (at the same time).** The current solution provides a general layout for a graph generation, but maybe users could benefit from seeing graph per context simultaneously, on the same page.

Chapter 4

Summary

The primary goal of the present thesis was to create a GSH visualization tool capable of handling multi-layered systems and provide context visualization at different levels of abstraction. The objectives stated on the research required produced graphs to be readable, have a hierarchical layout and properly handle collisions between elements. It was also required to have a context definition option.

The research is based on the existing knowledge and related works, practical and theoretical. The previous outcome of TUT bachelor thesis written by Maarja Raud under the supervision of Ants Torim is used as a foundation for GSH transformation functionality. The scientific paper-based Cytoscape.js library along with extensions supports graph generation. Existing GSH generation solutions and existing technologies were studied.

The main outcome of this work is designed and developed GSH visualization application which can be used in FCA field. The source code is made public in order to provide access to the tool to the greater auditory, to receive feedback from the greater number of people and to allow contributions.

The tool was validated by conducting a comparison with existing GSH visualization tools. The utility of the solution was tested and demonstrated on real use case. The validation results then were evaluated and discussed.

Based on the results of the research may conclude that stated problems were solved and requirements were satisfied.

The tool is publicly available and is expected to be applied in FCA analysis.

Public source code is also a possibility for new collaborators to easily get access to the project and participate in further development. The list of possible future works along with possible research directions was also provided in this work.

Bibliography

[1] Luis Elizondo. Why we moved from angular 2 to vue.js (and why we didn't choose react). https://medium.com/reverdev/why-we-moved-from-angular-2-to-vue-js-and-why-we-didnt-choose-react-ef807d9f4163, 2017. [Online; accessed 18-March-2019].

[2] Mark Monteleone. Building the security model with use case and class models. https://www.modernanalyst.com/Resources/Articles/tabid/115/ID/2181/Building-the-Security-Model-with-Use-Case-and-Class-Models.aspx, 20012. [Online; accessed 05-May-2019].

[3] Uta Priss. Linguistic applications of formal concept analysis. In *Formal Concept Analysis*, pages 149–160. Springer, 2005.

[4] Robert Godin and Petko Valtchev. Formal concept analysis-based class hierarchy design in object-oriented software development. In *Formal Concept Analysis*, pages 304–323. Springer, 2005.

[5] Robert Godin, Hafedh Mili, et al. Building and maintaining analysis-level class hierarchies using galois lattices. In *OOPSLA*, volume 93, pages 394–410, 1993.

[6] Robert Godin, Hafedh Mili, Guy W Mineau, Rokia Missaoui, Amina Arfi, and Thuy-Tien Chau. Design of class hierarchies based on concept,(galois) lattices. *Theory and Practice of Object Systems*, 4(2):117–134, 1998.

[7] Marianne Huchard and Hervé Leblanc. Computing interfaces in java. In *Proceedings ASE 2000. Fifteenth IEEE International Conference on Automated Software Engineering*, pages 317–320. IEEE, 2000.

[8] Marianne Huchard, A-D Seriai, Christelle Urtado, Sylvain Vauttier, A Al-Khlifat, et al. Concept lattices: a representation space to structure software variability. In *2014 5th International Conference on Information and Communication Systems (ICICS)*, pages 1–6. IEEE, 2014.

[9] U Priss. A graphical interface for document retrieval based on formal concept analysis. e. santos (ed.): Proc. of the 8th midwest artificial intelligence and cognitive science conf. Technical report, AAAI Technical Report CF-97-01, 66-70, 1997.

[10] Michel Liquiere and Jean Sallantin. Structural machine learning with galois lattice and graphs. In *ICML*, volume 98, pages 305–313, 1998.

[11] Sergei O Kuznetsov. Machine learning and formal concept analysis. In *International Conference on Formal Concept Analysis*, pages 287–312. Springer, 2004.

[12] Uta Priss. Formal concept analysis in information science. *Annual review of information science and technology*, 40(1):521–543, 2006.

[13] Bernhard Ganter, Gerd Stumme, and Rudolf Wille. *Formal concept analysis: foundations and applications*, volume 3626. springer, 2005.

[14] Bernhard Ganter and Rudolf Wille. *Formal concept analysis: mathematical foundations*. Springer Science & Business Media, 2012.

[15] Peter Eklund, Jon Ducrou, and Peter Brawn. Concept lattices for information visualization: Can novices read line-diagrams? In *International Conference on Formal Concept Analysis*, pages 57–73. Springer, 2004.

[16] Ants Torim. Galois sub-hierarchies used for use case modeling. In *CLA*, pages 21–32, 2013.

[17] Maarja Raud. Galois' sub-hierarchy graph generation tool. Bachelor thesis, Tallinn University of Technology, 2012.

[18] Kristo Aun. Tool for visualizing galois' sub-hierarchies. Bachelor thesis, Tallinn University of Technology, 2012.

[19] Rudolf Wille. Concept lattices and conceptual knowledge systems. *Computers & mathematics with applications*, 23(6-9):493–515, 1992.

[20] Petko Valtchev, David Grosser, Cyril Roume, and M Rouane Hacene. Galicia: an open platform for lattices. In *Using Conceptual Structures: Contributions to the 11th Intl. Conference on Conceptual Structures (ICCS'03)*, pages 241–254, 2003.

[21] Thomas Tilley. Tool support for fca. In *International Conference on Formal Concept Analysis*, pages 104–111. Springer, 2004.

[22] Anna Formica. Concept similarity in formal concept analysis: An information content approach. *Knowledge-Based Systems*, 21(1):80–87, 2008.

[23] Ken Peffers, Tuure Tuunanen, Marcus A. Rothenberger, and Samir Chatterjee. A design science research methodology for information systems research. *Journal of Management Information Systems*, 24(3):45–77, 2007. doi: 10.2753/MIS0742-1222240302. URL `https://doi.org/10.2753/MIS0742-1222240302`.

[24] Pivotal Software. Spring boot - java framework. `https://spring.io/projects/spring-boot#overview`, 2019. [Online; accessed 18-March-2019].

[25] Evan You. Vue.js - javascript framework. `https://vuejs.org/`, 2014-2019. [Online; accessed 18-March-2019].

[26] Quang Vinh Nguyen and Mao Lin Huang. Enccon: an approach to constructing interactive visualization of large hierarchical data. *Information Visualization*, 4(1):1–21, 2005.

[27] Stefan Berner, Stefan Joos, Martin Glinz, and Martin Arnold. A visualization concept for hierarchical object models. In *Proceedings 13th IEEE International Conference on Automated Software Engineering (Cat. No. 98EX239)*, pages 225–228. IEEE, 1998.

[28] Frank Van Ham and Jarke J van Wijk. Beamtrees: Compact visualization of large hierarchies. *Information Visualization*, 2(1):31–39, 2003.

[29] ВН Касьянов and ЕВ Касьянова. Визуализация информации на основе графовых моделей. *Научная визуализация*, 6(1):31–50, 2014.

[30] Emden R Gansner and Stephen C North. An open graph visualization system and its applications to software engineering. *Software: practice and experience*, 30(11):1203–1233, 2000.

[31] Weiwei Cui, Hong Zhou, Huamin Qu, Pak Chung Wong, and Xiaoming Li. Geometry-based edge clustering for graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1277–1284, 2008.

[32] Wassim CHEGHAM. Ngd: Angular dependencies graph. `https://github.com/compodoc/ngd/`, 2016. [Online; accessed 18-March-2019].

[33] Angular platform. `https://angular.io/`, 2010-2019. [Online; accessed 01-May-2019].

[34] Max Franz, Christian T Lopes, Gerardo Huck, Yue Dong, Onur Sumer, and Gary D Bader. Cytoscape. js: a graph theory library for visualisation and analysis. *Bioinformatics*, 32(2):309–311, 2015.

[35] Inc. npm. npm - javascript package manager. `https://www.npmjs.com/`, 2019. [Online; accessed 01-May-2019].

[36] Bootstrap team. Bootstrap 4 - front-end component library. `https://getbootstrap.com/`, 2019. [Online; accessed 01-May-2019].

[37] The Apache Software Foundation. Apache maven - software project management and compression tool. `https://maven.apache.org/`, 2002-2019. [Online; accessed 01-May-2019].

[38] Project lombok java library. `https://projectlombok.org/`, 2009-2019. [Online; accessed 01-May-2019].

[39] Gabriela Arévalo, Anne Berry, Marianne Huchard, Guillaume Perrot, and Alain Sigayret. Performances of galois sub-hierarchy-building algorithms. In *International Conference on Formal Concept Analysis*, pages 166–180. Springer, 2007.

[40] Anne Berry, Marianne Huchard, Amedeo Napoli, and Alain Sigayret. Hermes: an efficient algorithm for building galois sub-hierarchies. In *CLA: Concept Lattices and their Applications*, pages 21–32. Universidad de Malaga, 2012.

Appendix 1 - test formal context

Formal context used to generate GSH to demonstrate output of existing related systems (Section 1.1.2). Contains 6 objects and 25 attributes. Presented in three parts, as it is too long for single table.

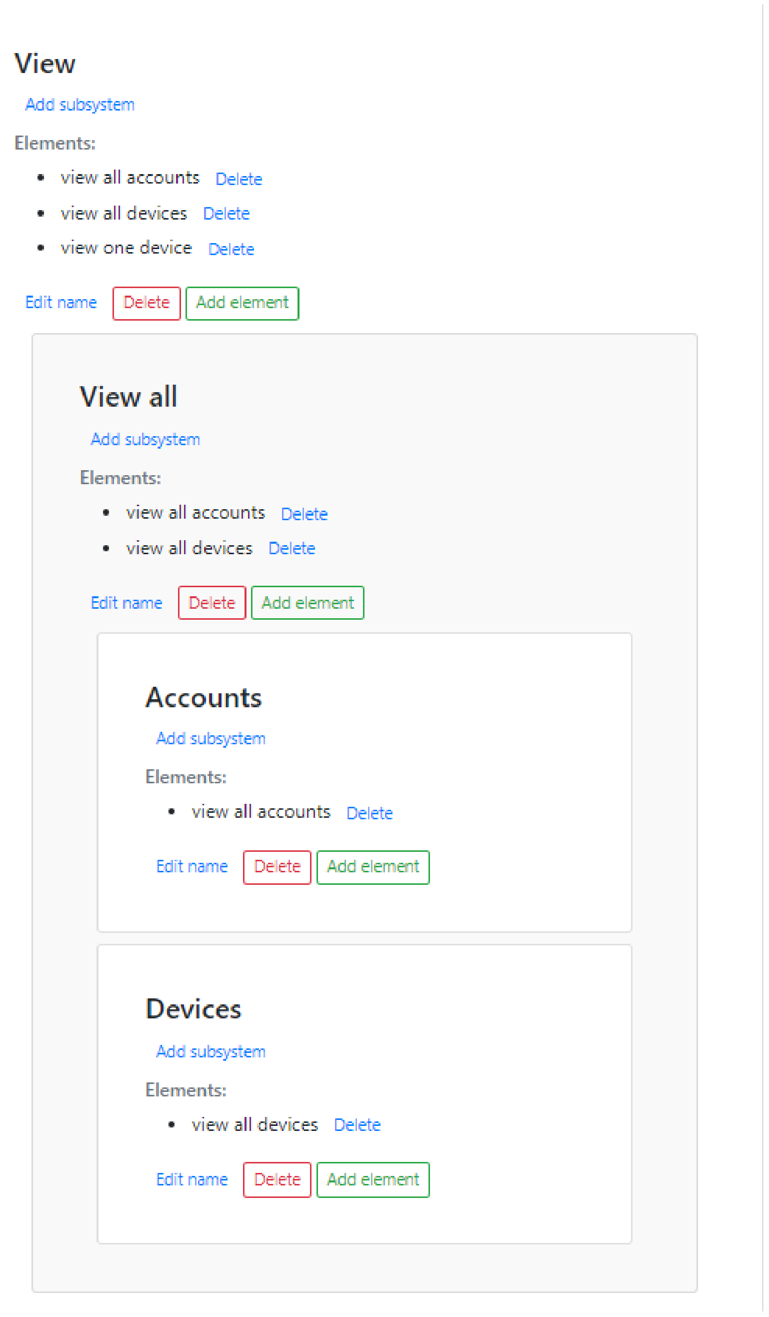
	attr_0	attr_1	attr_2	attr_3	attr_4	attr_5	attr_6	attr_7	attr_8
obj_0	1	1	1	1	1	1	0	0	0
obj_1	0	0	0	0	0	0	0	0	0
obj_2	0	0	0	0	0	1	1	0	0
obj_3	0	0	0	0	0	0	0	0	0
obj_4	0	0	0	0	0	0	1	1	1
obj_5	0	1	0	0	0	1	0	0	0

	attr_9	attr_10	attr_11	attr_12	attr_13	attr_14	attr_15	attr_16
obj_0	0	0	1	1	1	1	0	0
obj_1	0	0	1	1	1	1	1	1
obj_2	0	0	1	1	0	0	1	0
obj_3	0	0	0	1	0	0	1	0
obj_4	1	1	0	1	0	0	1	0
obj_5	0	0	1	1	1	1	1	1

	attr_17	attr_18	attr_19	attr_20	attr_21	attr_22	attr_23	attr_24
obj_0	0	1	0	0	0	0	0	0
obj_1	0	0	0	0	0	0	0	0
obj_2	1	1	1	1	0	1	0	1
obj_3	0	1	0	0	1	1	1	1
obj_4	0	1	0	0	1	1	1	1
obj_5	0	1	0	0	0	0	0	0

Appendix 2 - hierarchy of sub-systems

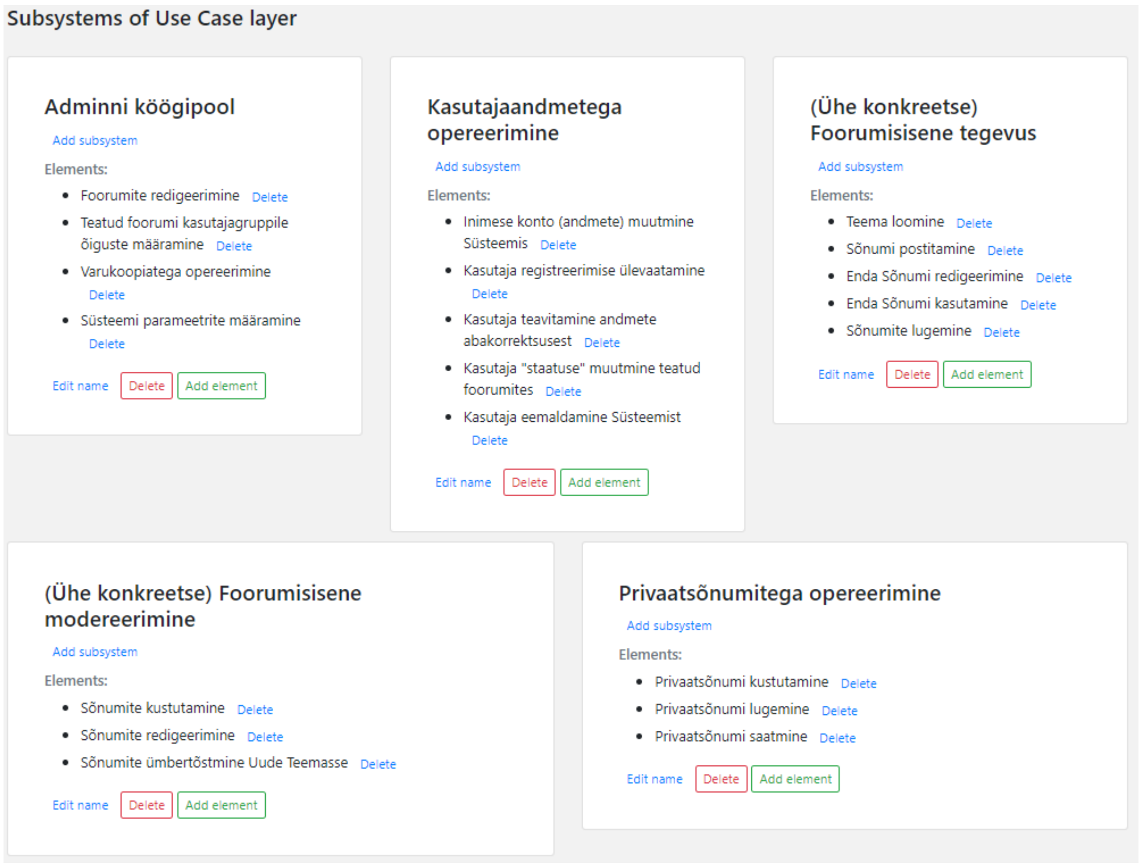
Screenshot from the application, that demonstrates how hierarchical sub-systems may look like.



Screenshot from the application - hierarchy of sub-systems (three levels)

Appendix 3 - hierarchy of sub-systems with no lower levels

Screenshot from the application, that demonstrates how the hierarchy of sub-systems with no lower levels may look like.



Subsystems of Use Case layer

Adminni köögipool

Add subsystem

Elements:

- Foorumite redigeerimine Delete
- Teatud foorumi kasutajagruppile õiguste määramine Delete
- Varukoopiatega opereerimine Delete
- Süsteemi parameetrite määramine Delete

Edit name Delete Add element

Kasutajaandmetega opereerimine

Add subsystem

Elements:

- Inimese konto (andmete) muutmine Süsteemis Delete
- Kasutaja registreerimise ülevaatamine Delete
- Kasutaja teavitamine andmete abakorrektsusest Delete
- Kasutaja "staatuse" muutmine teatud foorumites Delete
- Kasutaja eemaldamine Süsteemist Delete

Edit name Delete Add element

(Ühe konkreetse) Foorumisisene tegevus

Add subsystem

Elements:

- Teema loomine Delete
- Sõnumi postitamine Delete
- Enda Sõnumi redigeerimine Delete
- Enda Sõnumi kasutamine Delete
- Sõnumite lugemine Delete

Edit name Delete Add element

(Ühe konkreetse) Foorumisisene modereerimine

Add subsystem

Elements:

- Sõnumite kustutamine Delete
- Sõnumite redigeerimine Delete
- Sõnumite ümbertõstmine Uude Teemasse Delete

Edit name Delete Add element

Privaatsõnumitega opereerimine

Add subsystem

Elements:

- Privaatsõnumi kustutamine Delete
- Privaatsõnumi lugemine Delete
- Privaatsõnumi saatmine Delete

Edit name Delete Add element

Screenshot from the application - hierarchy of sub-systems (upper level only)