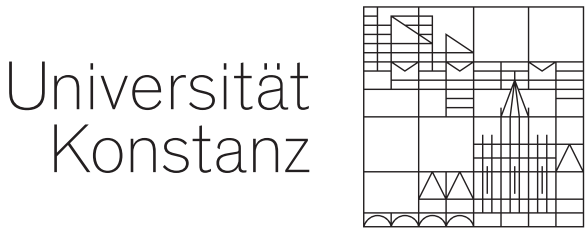
Minimization of Stochastic Dynamical Systems

A Comparison of Models & Methods

M.Sc. Project Report

by

Fabian Klopfer



Faculty of Sciences
Department of Computer and Information Science
Modelling of Complex Self-Organizing Systems Group

Supervisor: Dr. Stefano Tognazzi
Professor: Prof. Dr. Tatjana Petrov

Konstanz, 2020

Contents

1. Introduction

2. Models

3. Reductions & Minimization

4. Evaluation

Bibliography

[1] Steven Abney, David McAllester, and Fernando Pereira. "Relating probabilistic grammars and automata". In: *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*. 1999, pp. 542–549.

[2] Giovanni Bacci et al. "On the metric-based approximate minimization of Markov Chains". In: *Journal of Logical and Algebraic Methods in Programming* 100 (2018), pp. 36–56.

[3] Andrea Bobbio and Kishor S. Trivedi. "An aggregation technique for the transient analysis of stiff Markov chains". In: *IEEE Transactions on computers* 9 (1986), pp. 803–814.

[4] François Boulier et al. "Towards an automated reduction method for polynomial ODE models of biochemical reaction systems". In: *Mathematics in Computer Science* 2.3 (2009), pp. 443–464.

[5] George Edward Briggs and John Burdon Sanderson Haldane. "A note on the kinetics of enzyme action". In: *Biochemical journal* 19.2 (1925), p. 338.

[6] Janusz A Brzozowski. "Canonical regular expressions and minimal state graphs for definite events". In: *Mathematical theory of Automata* 12.6 (1962), pp. 529–561.

[7] Peter Buchholz. "Bisimulation relations for weighted automata". In: *Theoretical Computer Science* 393.1-3 (2008), pp. 109–123.

[8] Peter Buchholz. "EXACT AND ORDINARY LUMPABILITY IN FINITE MARKOV CHAINS". In: 1994.

[9] RG Bukharaev. "Probabilistic automata". In: *Journal of Soviet Mathematics* 13.3 (1980), pp. 359–386.

[10] Luca Cardelli et al. "Maximal aggregation of polynomial dynamical systems". In: *Proceedings of the National Academy of Sciences of the United States of America* 114 38 (2017), pp. 10029–10034.

[11] Charles Francis Curtiss and Joseph O Hirschfelder. "Integration of stiff equations". In: *Proceedings of the National Academy of Sciences of the United States of America* 38.3 (1952), p. 235.

[12] Pedro R DArgenio and Joost-Pieter Katoen. "A theory of stochastic systems part I: Stochastic automata". In: *Information and computation* 203.1 (2005), pp. 1–38.

[13] Pedro R DArgenio and Joost-Pieter Katoen. "A theory of stochastic systems. Part II: Process algebra". In: *Information and Computation* 203.1 (2005), pp. 39–74.

[14] Peter Deuflhard and Jörg Heroth. "Dynamic dimension reduction in ODE models". In: *Scientific computing in chemical engineering*. Springer, 1996, pp. 29–43.

[15] Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of weighted automata*. Springer Science & Business Media, 2009.

[16] Manfred Droste and Dietrich Kuske. "Weighted automata". In: *Handbook AutoMathA* (2013).

[17] Albert Einstein. "On the theory of the Brownian movement". In: *Ann. Phys* 19.4 (1906), pp. 371–381.

[18] Kousha Etessami. "Analysis of probabilistic processes and automata theory". In: *Automata: from Mathematics to Applications* (2016).

[19] Bernhard C Geiger et al. "Optimal kullback–leibler aggregation via information bottleneck". In: *IEEE Transactions on Automatic Control* 60.4 (2014), pp. 1010–1022.

[20] Stuart Geman and Mark Johnson. "Probabilistic grammars and their applications". In: *International Encyclopedia of the Social & Behavioral Sciences* 2002 (2002), pp. 12075–12082.

[21] G Grinstein, C Jayaprakash, and Yu He. "Statistical mechanics of probabilistic cellular automata". In: *Physical review letters* 55.23 (1985), p. 2527.

[22] Jack K Hale et al. *Introduction to functional differential equations*. Vol. 99. Springer Science & Business Media, 1993.

[23] Thomas Henzinger, Barbara Jobstmann, and Verena Wolf. "Formalisms for Specifying Markovian Population Models". In: vol. 22. Sept. 2009. DOI: 10.1007/978-3-642-04420-5_2.

[24] John Hopcroft. "An n log n algorithm for minimizing states in a finite automaton". In: *Theory of machines and computations*. Elsevier, 1971, pp. 189–196.

[25] Tao Jiang and Bala Ravikumar. "Minimal NFA problems are hard". In: *SIAM Journal on Computing* 22.6 (1993), pp. 1117–1141.

[26] Tsunehiko Kameda and Peter Weiner. "On the state minimization of nondeterministic finite automata". In: *IEEE Transactions on Computers* 100.7 (1970), pp. 617–627.

[27] Michael N Katehakis and Laurens C Smit. "A successive lumping procedure for a class of Markov chains". In: *Probability in the Engineering and Informational Sciences* 26.4 (2012), pp. 483–508.

[28] Stefan Kiefer and Björn Wachter. "Stability and complexity of minimising probabilistic automata". In: *International Colloquium on Automata, Languages, and Programming*. Springer. 2014, pp. 268–279.

[29] Stefan Kiefer et al. "On the Complexity of Equivalence and Minimisation for Q-weighted Automata". In: *Logical Methods in Computer Science* 9 (2013).

[30] Peter Kunkel and Volker Mehrmann. *Differential-algebraic equations: analysis and numerical solution*. Vol. 2. European Mathematical Society, 2006.

[31] Wilhelm Kutta. "Beitrag zur naherungsweisen integration totaler differentialgleichungen". In: *Z. Math. Phys.* 46 (1901), pp. 435–453.

[32] Pierre-Simon Laplace. *Philosophical Essay on Probabilities: Translated from the fifth French edition of 1825*. Vol. 13. Springer Science & Business Media, 1998.

[33] Cristopher Moore and James P Crutchfield. "Quantum automata and quantum grammars". In: *Theoretical Computer Science* 237.1-2 (2000), pp. 275–306.

[34] Isaac Newton. "Methodus fluxionum et serierum infinitarum". In: *Opuscula mathematica, philosophica et philologica* 1 (1774), p. 1774.

[35] Bernt Øksendal. "Stochastic differential equations". In: *Stochastic differential equations*. Springer, 2003, pp. 65–84.

[36] Stephan Olariu and Albert Y Zomaya. "Cellular Automata, PDEs, and Pattern Formation". In: *Handbook of Bioinspired Algorithms and Applications*. Chapman and Hall/CRC, 2005, pp. 291–302.

[37] Alan V Oppenheim. *Discrete-time signal processing*. Pearson Education India, 1999.

[38] Steven A Orszag. "Numerical methods for the simulation of turbulence". In: *The Physics of Fluids* 12.12 (1969), pp. II–250.

[39] Azaria Paz. *Introduction to probabilistic automata*. Academic Press, 2014.

[40] Tatjana Petrov. "Markov chain aggregation and its application to rule-based modelling". In: *CoRR* abs/1812.09774 (2018). arXiv: 1812.09774. URL: http://arxiv.org/abs/1812.09774.

[41] Linda Petzold. "Differential/Algebraic Equations Are Not ODEs". In: *SIAM J. Sci. Stat. Comput.* 3.3 (Sept. 1982), pp. 367–384. ISSN: 0196-5204. DOI: 10.1137/0903023. URL: https://doi.org/10.1137/0903023.

[42] Michael O Rabin. "Probabilistic automata". In: *Information and control* 6.3 (1963), pp. 230–245.

[43] John Rhodes, Chrystopher L Nehaniv, and Morris W Hirsch. *Applications of automata theory and algebra: via the mathematical theory of complexity to biology, physics, psychology, philosophy, and games*. World Scientific, 2010.

[44] Jean-Pierre Richard. "Time-delay systems: an overview of some recent advances and open problems". In: *Automatica* 39.10 (2003), pp. 1667–1694. ISSN: 0005-1098. DOI: https://doi.org/10.1016/S0005-1098(03)00167-5. URL: http://www.sciencedirect.com/science/article/pii/S0005109803001675.

[45] Claus Scheiderer. *Script: Linear Algebra I*. Department of Mathematics and Statistics, University of Konstanz, 2017.

[46] Uwe Schöning. *Theoretische Informatik-kurz gefasst*. BI-Wiss.-Verlag, 1992.

[47] Marcel Paul Schützenberger. "On the definition of a family of automata". In: *Information and control* 4.2-3 (1961), pp. 245–270.

[48] C. E. Shannon and J. McCarthy. *Automata Studies.(AM-34)*. Vol. 34. Princeton University Press, 1956.

[49] George F Simmons. *Differential equations with applications and historical notes*. CRC Press, 2016.

[50] Ana Sokolova and Erik P De Vink. "Probabilistic automata: system types, parallel composition and comparison". In: *Validation of Stochastic Systems*. Springer, 2004, pp. 1–43.

[51] Ana Sokolova and Erik P De Vink. "Probabilistic automata: system types, parallel composition and comparison". In: *Validation of Stochastic Systems*. Springer, 2004, pp. 1–43.

[52] Mariëlle Stoelinga. "An introduction to probabilistic automata". In: *Bulletin of the EATCS* 78.176-198 (2002), p. 2.

[53] Stephen Strogatz. "Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering (studies in nonlinearity)". In: (2001).

[54] Robert Tarjan. "Dept-first search and linear graph algorithms". In: *Siam Journal on Computing - SIAMCOMP* 1 (Jan. 1997).

[55] Antti Valmari and Giuliana Franceschinis. "Simple O (m logn) time Markov chain lumping". In: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer. 2010, pp. 38–52.

[56] Nicolaas Godfried Van Kampen. *Stochastic processes in physics and chemistry*. Vol. 1. Elsevier, 1992.

[57] Vito Volterra. "Theory of functionals and of integral and integro-differential equations". In: (1959).

[58] John Von Neumann. *Mathematical foundations of quantum mechanics: New edition*. Princeton university press, 2018.

[59] John Von Neumann, Arthur W Burks, et al. "Theory of self-reproducing automata". In: *IEEE Transactions on Neural Networks* 5.1 (1966), pp. 3–14.

[60] Thorsten WiSSmann et al. "From Generic Partition Refinement to Weighted Tree Automata Minimization". In: *arXiv preprint arXiv:2004.01250* (2020).

[61] Daniel Zwillinger. *Handbook of differential equations*. Vol. 1. Gulf Professional Publishing, 1998.

A. Software Requirement Specification

A.0.1. Introduction

A.0.1.1. Purpose

The purpose of this Software Requirements Specification is to describe the features, constraints and demands of a Zoo Management Tool in detail. This document is intended for both the stakeholders and the developers of the system and will be proposed to the zoo director Dr. Susan Seuss and her Staff.

A.0.1.2. Scope

This Software system shall implement a Zoo Management Tool, that assists the zoo-keepers in ordering feed, managing and visualizing the feeding costs of each animal , the administration in managing the budgets, preferenced food dealers, communication, staff and working hours and the system itself.

A.0.1.3. Definitions, Acronyms, Abbreviations

word	shortform	meaning
database	db	a facillity to store and alter data
Operating System	os	the basic software on a machine
C Programming Language	C	a programming language, Windows, macOS, linux and may more os are based on it
functions and methods	-	programmed descriptions of the working steps of a reoccuring task on information
user	-	person, that is going to use the here described software.
access rights	-	the permission rights on data or the alteration of data
usergroup	-	a category for users with the same access rights
...	...	...

A.0.1.4. Overview

In the second chapter several conditions, assumptions and circumstances will be mentioned, that help charachterizing the software's special use case. In the thrid chapter the concrete requirements are listed.

A.0.2. Overall description

A.0.2.1. Product Perspective

The product shall support three main tasks of the daily work in the zoo: 1. Budget management, 2. Feed management and 3. Scheduling management. This shall be done by a database, that shall save related information and functions and methods, that operates on this data.

A.0.2.2. Product Functions

The system shall store information on the budget and take care of balancing each month. It shall support the zoo staff in communicating request for additional funds, by providing forms, notifications and historical information on the requests. The system shall store information on available feeds, their expiration date, orders and invoices. It shall also store information about employees and schedules of holidays and working hours.

A.0.2.3. User Characteristics

The users are the staff of a zoo. Therefore standard user will not have advanced technical knowldege, but they are able to visit websites to order food and to read from the graphic user interface of a tablebased database.
There are three different type of users:

- Zoo keepers: responsible for animals and their feed, including the ordering of new feed. They have the right to view animal budgets, they have the right to view and alter the feed information. They don't have the right to view employee information or other budget information than animal budget information.

- Secretary: Cares about scheduling working hours, vacation, the employee budget, delivery issues and probably other occuring problems. Therefore she needs access on Employee information and the employee budget, orders, invoices, and eventually more. So there shall be a usergroup, that is easy adjustable, so that the secretary can be granted temporary rights for things she only need to handle once or to change her permanent access and alter rights quickly. This usergroup needs to be more secure.

- Zoo director: The administrator of the zoo and the future system. Needs full access and altering rights; has to be most secure.

A.0.2.4. Constraints

- Security critical accounts for usergroups 2 and 3, due to personal and financial critical data.

A.0.2.5. Assumptions and Dependencies

The tablets for the software to be run on are present and provide an operating system, that implements a messaging system and is compatible to the C programming language (and by that C based languages).

A.0.3. Specific Requirements

A.0.3.1. External interfaces

Tablets and their os, a fitting database management system (details omitted, no implementation/concrete facts about this)

A.0.4. Functional requirements

1. Order- and feed-management
    1. The system shall highlight entries that passed their expiration date.
    2. The system shall be able to check if a planned orders invoice total will exceed the monthly budget of an animal.
    3. The system shall only deduct the invoice total from the budget if an order arrived.
    4. The system shall provide a arrived checkbox for each order to mark it as arrived with a date stamp.
    5. If an order does not arrive in time, the system shall mark it "overdue" and highlight it.
    6. The system has to be implemented so, that the list of authorized feed dealers can be edited by a user with sufficient rights in the system.
    7. The system shall provide a request for additional funds form for the corresponding user if the invoice total of an order would exceed budget or if the planned payment of a temporary worker would exceed the budget.
2. Budget-management
    1. The system shall be so implemented, that at the end of a month only the spent amount of the budget shall be deducted from the total zoo budget.
    2. The system shall reserve the sum of the animal budgets from the overall budget.
    3. The system shall make it possible to alter the monthly budget and the monthly reserved budgets. A user with sufficient rights in the system shall be able to write the new amounts and confirm them with its log-in data. The person or persons that are concerned with the altered budget shall be informed by the system.
    4. The system shall include a graphical representation of the total monthly budget, money already spent in the current month, reserved monthly sub-budgets (such as budgets for feed or salaries and wages) and money not spent or reserved for this month.
    5. The system shall provide a graphical representation of: total animal budget of the month, amount spent already, amount reserved for pending deliveries, amount not spent for each animal.
    6. The system shall balance the account monthly and carry the result to the next month.
    7. The system shall be able to store and print invoices.
    8. The system shall highlight the balance green if it's positive and red if it's negative. If it's negative the system shall issue a warning.
    9. The system shall send a message to the zoo director if a request for a budget increase is submitted; there shall be a link to the corresponding budget, the stock of the corresponding animals feed and an average cost per order.
    10. The system shall include an overview on the budget spanning the last 6 month. This overview shall include: money spent for every month, average money spent, number of budget increase requests, number of requests granted, number of requests denied, links to the corresponding statements. This overview shall be generated and linked in the message to the zoo director for each request that is submitted.
    11. If the zoo director grants an increase request, the system shall make it possible to enter the amount by which the budget is increased. This amount shall be added to the current months reserve. A message shall be send to the applicant, containing the information that the request was granted and the amount by which the budget was increased. In case of the applicant being a zookeeper a message shall be send to all zookeepers that are responsible for the animal or animals affected by this request. The affected budget shall be increased. In the overview this shall be marked as additional funds granted on request.

12. If the request is denied, a message that contains a short statement written by the zoo director shall be send to the applicant.

3. Schedule-management
    1. The system shall, if a temporary worker is needed, check if there is enough money left in the employee budget.
    2. If there is a delay in delivery the system shall inform the secretary.
    3. The system shall be able to sort all personnel by tasks, working hours and availability.
    4. The system shall warn the user if one tries to include a worker into the schedule during their planned holiday.
    5. If the secretary is able to resolve the problem with the food dealer, the system shall send a message to all zoo keepers of the corresponding animal, including a statement and if available the new delivery date. The new delivery date shall be updated to the corresponding order and marked as new delivery date.
    6. The system shall inform the zoo director if a temporary worker needs to be hired more than 40 hours in one month.
    7. If an user wants to plan a vacation, this user has to get a permission by the secretary. The system shall check for conflicts with other planned vacation of personnel with the same task or tasks and if there are enough unused holidays. If the secretary grants the request for holidays, the system shall send a message to the employee and the planned vacation shall be entered into the employee information. If she does not grant the request, a notification with a short statement shall be send to the applicant by the system. In the employee information the system shall remark that a request for holidays was denied with a link to the request or requests.

A.0.5. Datebase requirements

1. The system shall provide a separate budget for every animal.
2. The system shall store information about the available feeds including their quantity and their expiration dates.
3. The system shall provide the information how much of the monthly budget was already spent on an animal.
4. The system shall store data on the budget, containing how much was spent, for what and when, how much was earned, by what and when, how much was reserved, for what and when, money left/balance.
5. The system shall store employee information containing the following: salary or wage, working hours, unused holidays, planned holidays, requested holidays, denied holiday requests, availability for overtime and in case of temporary workers availability and hours worked in the current month, hours worked total and the tasks that they are able to fulfill.
6. The system shall be able to store orders, containing its invoces and an expected delivery date.
7. Orders shall be deleteable.

A.0.6. Software System requirements

1. Maintainability
2. Security
3. Stability
4. Cost efficience

A.0.7. Auditory multi task model

The audio system functions as follows:

1. When human speech is detected, chunks of audio (e.g. 100ms long chunks) are sent consecutively to a speech recogniser (e.g. Google Speech). The speech recogniser sends back an intermediate result. When the human voice has finished, the speech recogniser will send a final result.
2. When the human voice is detected and the audio chunks are arriving, each audio chunk should be sent to the speech recogniser and the multi task model. They both compute intermediate results, which are documented below. A single ROS message will be sent with all of the intermediate results: e.g. text, confidence, language, voice_id, voice_expressions and voice_acitivities.
3. When the human voice has finished, a final audio chunk is sent to the speech recogniser and multi task model, which both compute the final result. A single ROS message will be sent with all of the final results: e.g. text, confidence, language, voice_id, voice_expressions and voice_acitivities.
4. We assume 1 speaker.

The target properties (those computed by multi model include Properties, Voice ID, Voice Expressions, Voice Acitvities), in addition to what is computed by the speech recogniser:

Recognised speech: (will je Google Speech to start with):

- Text: recognised speech, converted to text.
- Confidence: confidence of result.
- Language: language of recognised speech speech.

Properties:

- List of property names (e.g. age, gender, ethnicity, volume)
- List of property values
- List of confidences

Voice ID (e.g. 64-dimensional)

- Vector (feature vectors are input independent as they extract the degree of certain traits, like the pitch of the voice. Further sound may be seen as an image (just draw the frequencies). If one does fourier analysis one extracts distinct frequency bands (an example for hand-crafted features, here math. implied)
- Confidence

Voice Expressions

- List of voice expression names (e.g. happy, sad - the models used could be customised, e.g. we could use emotion, valence, arousal etc this is data st dependent etc)
- List of confidences

Voice activities (could also assume you have an image of the face for this one)

- List of voice activity names (e.g. speaking, laughing, yawning, coughing)
- List of confidences

Possible datasets for training include:

- Voice ID:
- Voice Expression:
- Voice acitvities:

How have people solved these before:

- Voice ID:
- Voice Expression:
- Voice activity: e.g.

B. Software Design Document

B.0.1. Introduction

B.0.1.1. Purpose

This document is written, to document the proposed architecture of a zoo management system. The document is inteded for the developers implementing the system, the persons deploying the final system and the other stakeholders to see the process and decide, if the developed system is according to the specifications.

B.0.1.2. Scope

The System developed is a zoo management system, that assists the zoo keepers in ordering food, managing and visualizing the feed cost of each animal, the administrators in managing budgets, preferred feed dealers and the staff. The software requirements specification is already present.

B.0.1.3. Definitions, Acronyms, Abbreviations

term	**short form**	**description**
Java	-	A portable programming language
Java Virtual Machine	JVM	The software system nessecary to run java applications.
Database Management System	DBMS	A software system used to manage, create and maintain databases.
relational DBMS	-	DBMS storing data in tables referencing each other.
Transport Control Protocol / Internet Protocol	TCP/IP	Low level Protocols to transfere data over the internet.
Secure Hypertext Transfer Protocol	HTTPS	The standard protocol to transfer data over the internet in a secure fashion.

B.0.1.4. References

1. SRS Document
2. Enumerated Requirements Document

B.0.1.5. Overview

The proposed architecture will be shown from a static point of view, eg. the decomposition into subsystems and their structure. Then the final deployment will be shown. This includes the distribution of the software across the different hardware used. Furthermore the data management is visible. Therefore the database design is modeled and the used DBMS is mentioned. The next part contains a description of the security features of the system and how the users are managed. The dynamic aspects of the system are following. This contains the description of interfaces between components, the behavior of the system and information flow. Finally, used design patterns are mentioned.

B.0.2. Current Architecture

There is no current architecture. It is assumed, that there is a messaging system availible, to send messages to specific users/devices.

B.0.3. Proposed Software Architecture

B.0.3.1. Overview

The Zoo management system is a client - server system. It is composed out of the following components.

Client The client module is the complete system running on the client computer. It bundles the UI, navigation and can obtain an access token for the inner components.

Server The server module is the system running on the server computer. It hosts the database and provides user authentification capabilities.

FeedManager (FM) The FM system coordinates all requests concerning the supply, storage and management of feeds and the according feed dealers. It is mainly used by zoo keepers for buying feeds and checking the current quantity and expiration date of stock feeds.

EmployeeManager (EM) The EM system is mainly used by the secretary to create employee schedules, organize temporary workers, check the vacation requests of employees and manage the associated budget.

BudgetManager (BM) The BM system organizes the budget of the zoo. It is used to generate reports, get an overview of the current budget and provide information to prevent negative balances. It is mainly used by the zoo director to organize the subbudgets for feeds and employees as well as granting or dening requests for additional funding created by other employees.

UserManager (UM) The UM system is used to create and manage user accounts of the system. It is exclusively used by the zoo director.

Messaging subsystem (MSS) The MSS is used by the system to deliver messages to special user accounts. It is an external system, which is used by the zoo management system.

MessageManager (MM) The Message manager is a client component, which is used to display messages that are received from the MSS.

AuthUtility (AU) The AU is a compontent running on the server. It is the only way to access the system internals. This component checks, if provided login credentials are valid, and if the access level of the user is high enough.

Database subsystem (DBMS) The DBMS is the subsystem used for persistent data storage. It is used as a transaction based system.

B.0.3.2. Subsystem decomposition

Client The client component contains the main initialisation code. It provides a host window for the other components to draw their UI and navigational elements to switch the used component. In addition the component is used to obtain an access token from the server, which allows other components to comunicate with the server.

Server The server component contains the initialisation code to start the DBMS and routes incoming connections to the AuthUtility.

Manager Components The different manager components are implemented in a similar fashion: By using the *AbstractFactory* Pattern to get instances. The following two models figure and emphasis this pattern and the implementaion style.

The *BudgetManager* is instantiated by the *BudgetManagerFactory*. The budgets are modeled as *CompositeBudget*s and *SubBudget*s, both supporting a specific set of operations. In addition animal implements the *Budget* interface, because each *Animal* has an associated budget.

MessagingSubSystem The messaging susbsystem allows the zoo management system to send messages to specific user accounts and devices. The system has to guarantee the delivery of the messages. The architecture isn't specified any further, because the MSS is an external system and therefore not part of this document.

AuthUtility The AuthUtility is used to access the database. For every request the AuthUtility checks, if the login credentials are valid and not expired. Then the privilege level for the requested action is checkt. If the user has enough permissions the request is passed to the DBMS. The answer is returned to the client.

DatabaseManagementSystem The DBMS used is MariaDB. It is a high performance relational database management system. It is used, because it is a open source software and therefore free of costs. In addition it is widely used and tested. Therefore it is reliable and it may be assumed, that it is supported for the entire lifetime of the zoo management system.

B.0.3.3. Hardware/Software mapping

Figure shows the planned hardware software mapping of the zoo management system. The System is divided into a server and a client part. The server part will run on a server computer, availiable 24/7. The server is responsible for user authentication and the storage of data. The client side consists of tablets and personal computers. These will run the client application, which is used to contact the server via a HTTPS (on top of TCP/IP). The client application requires the Java Runtime Envirement to be installed on personal computers and tablet clients. The client provides the user interface and uses the messaging subsystem to deliver and receive messages to/from users of the system. The hardware allocation of the MSS is not part of this document.

B.0.3.4. Persistent data management

For persistent data management on the server side the open source relational DBMS MariaDB is used. Data is organized in tables according to the scheme in figure. Users have multiple activity logs associated. Each employee has exactly one user account for the system. For each employee, the associated vacation requests and a schedule is stored. Employees are workers of the zoo, so some data is shared with temporary workers (which are workers, too), for example name and wage/salary. For each worker there is possibly a WorkerFundRequest, if the budget was too small to hire the worker. This, in turn, is a specialization of a general FundRequest, which is associated with a given budget. Each budget consists of multiple subbudgets and may contain FundRequests. Another possible FundRequest is the FeedFundRequest, which is issued, if an order exceeds the feed budget. These requests and the associated orders are stored and linked. An order can be delayed and the delivery may be updated, so the possible DeliveryUpdates are stored. The order itself contains one or more Feed offerings. These offerings are stored as an association between FeedDealer and Feed, because the FeedDealer can make an offer for a feed. If a FeedDealer is removed from the system, all the associated feed offerings not contained in an order are deleted too. In addition the system stores all animals and the feeds they are eating. If an animal is removed from the system, the feeds consumed only by this one animal are deleted. If a feed is deleted, the associated offerings are removed, if they aren't contained in any order.

On the client side no DBMS is neseccary, because all used data is loaded at runtime. The only presistent data is the username, which is stored for the next usage of the system.

B.0.3.5. Access control and security

Network location restrictions The first layer of security is, to grant access to the system only out of the local network in the zoo. If this is too restrictive, it may be dropped.

Network protocol restrictions The client may only connect to the server using a secure HTTPS connection. This require the server to have a valid SSL certificate. The server will drop every connection attempt using insecure protocols.

User account restrictions The next layer is the AuthUtility. Each user of the system has a user account with a unique user ID and a password. These accounts are managed by the zoo director. When a user tries to contact the application server, he has to provide a valid access token and a valid user account id. The access token is obtained by sending the login credentials to the authentification server. If they are valid an access token is returned. This token has only limited validitiy and has to be obtained again, if timed out. To restrict access for zoo keepers and the secretary, every user account has an integer value associated, which represents the users access level. Each module can check the current access level of the user to determine which functionality is provided.

Storage policies The user passwords aren't stored in plain text. They are hashed using a state of the art hash funktion (PBKDF2 with iteration count larger than 20000) and stored together with the used hash function, salt, and function parameters (iteration count). Because a lot of people use the same password for different services, this prevents potential intruders from taking the password list and using the passwords on other web services or obtaining passwords from other services and using them for the zoo management system.

Logging As an additional security feature the system logs the account accessing the system for every access. These logs are visible for the zoo director.

B.0.3.6. Global software control

Startup behaviour **Server:** When the server machine is booted, the operating system automatically invokes a startup handler of the system. This handler will boot up the database, the AuthUtility and then it will enable the request handling.
Client: The client application starts on user demand. The client will show a login form for the user. When the user entered his login data the client contacts the server to obtain an access token. The client creates different instances of *UIComponentFactory* for each UI component. The components are initialized with the access token and these factory objects.

Interfaces The `MessagingService` is provided by the MSS. It provides a possibility to notify users with messages. If the notifyUser function is invoked, it is ensured, that the user is notified and the message is stored.

```
Interface MessagingService {
  public void sendMessage(string sender,
      string accesstoken, string recipient, Message message) {
    require accessValid(username, accessToken);
    require messageHandle != null;
    require message.length > 0;
    ensure getMessages(recipient, database.getUser(recipient)
        .getAccessToken()).contains(message);
  }
```

```
  public Message[] messages getMessages(string username, string accessToken) {
    require accessValid(username, accessToken);
    require messageHandle != null;
    ensure messages != null;
  }

  public void messageRead(String username, string accessToken Message message) {
    require accessValid(username, accessToken);
    require message != null;
    require getMessages(username, accessToken).contains(message);
    ensure !getMessages(username, accessToken).contains(message);
  }
}
```

The `AccessWebService` interface provides three functions: *getAccessToken*, *accessValid* and *access*. *getAccessToken* is used to obtain access tokens. *accessValid* checks, if the provided access token is valid. This function is used by the MSS to authenticate users. The function *access* is used to execute operations on the database. In before a authentification check is made and it is tested if the user is allowed to execute the given operation.

```
Interface AccessWebService {
  public string accesstoken getAccessToken(string username, string password) {
    require username.length > 0 && password.length > 0;
    ensure database.getUser(username).accesstoken == accesstoken &&
    database.getUser(username).expirationDate;
    ensure containsLogEntry(user, system.getCurrentDate());
  }

  public boolean valid accessValid(string username, string accesstoken) {
  require database.existsEntry(username);
  ensure valid == database.getUser(username).isIsValidToken(accesstoken);
  }

  public Response r access(string username,
                           string accesstoken, Operation op) {
    require database.entryExists(username);
    ensure iff accessValid(username, accesstoken) then
      iff database.getUser(username).isLocked() then
        operation == new PasswordChangeRequiredResponse();
      else
        iff database.getUser().getPrivilegeLvl() > getPrivilegeLvl(operation) then
          operation is executed and r is the response of the operation;
        else
          r == new AccessDeniedResponse();
    ensure if accesstoken is not valid operation is not executed and r is null;
  }
}
```

Sequences table

Sequence Name	**accessValid**
Preconditions	- username is contained in the database - accesstoken is not null
Event order	0. initial call 1. AuthUtility requests the given User from the DB 2. The DB allocates and populates a User Object 3. The BD returns the User Object 4. AuthUtility ask the User object, if the token is valid 5. The User Object returns a boolean 6. This boolean is passed to the client
Postconditions	- the database is not modified
Quality requirements	- The accessValid call shall finish within at most 3 seconds.

Sequence Name	**getAccessToken**
Preconditions	- username is contained in the database - password is not null
Event order	0. initial request 1. AuthUtility requests the given User from the DB 2. The DB allocates and populates a User Object 3. The BD returns the User Object 4. The AuthUtility generates the password hash using HashImpl 6. The AuthUtility gets the original password hash from the User object 8. The hashes are compared 9. The hashes were equal so an access token is requested from the user object 10. The user object checks, if the currently active access token is valid 11. The token wasn't valid, so a new token is generated 12. The new Token is saved in the Database 13. Now the access token is valid and therefore returned to the AuthUtility 14. From there on it is passed to the callee 15. The password was false so null is returned
Postconditions	- the returned access token is valid - the returned access token is saved persistently in the DB
Quality requirements	- The getAccessToken call shall finish within at most 3 seconds.

Sequence Name	**access**
Preconditions	- username is contained in the database - access token is not null - the operation is not null
Event order	0. initial request 1. AuthUtility requests the given User from the DB 2. The DB allocates and populates a User Object 3. The BD returns the User Object 4. it is checked, if the access token is valid (using the User object) 6. the token was valid, so the access level of the user is checked 8. the access level of the operation and the user are compared 9. the user is allowed to execute the operation, so the operation is send to the DB 10. The DB Response is passed to the AuthUtility 11. The Response is passed to the callee 12. the user had no sufficient access permissions so null is returned to the callee 13. the access token was not valid, so null is returned
Postconditions	- the database is not modified
Quality requirements	- The accessValid call shall finish within at most 3 seconds.

B.0.3.7. Boundary conditions

1. MSS exists already.
2. The zoo has a sufficiently fast network connection availible to use the system without maintaining full disk caches.

B.0.3.8. Patterns

AbstractFactory The abstract factory pattern is used, to allow the client component to initialize the UI without knowing the exact implementation. The client component provides different interfaces (figure) which are implemented by UI components (for example figure and figure).

Composition The composition pattern is used by the BudgetManager (figure). The interface *Budget* specifies common operations possible on composed budgets and atomic budgets. It is implemented in different subclasses for example *SubBudget* and *CompositeBudget*. This is useful, because the controller doesn't have to know if a budget is composed out of different budgets or if it is an atomic budget. This simplifies the implementation of the overview screen and other parts using the budget.

B.0.4. Changelog

1. enumerated requirements (Enumerated Requirements Document)
2. language change to Java (C is impractical for client applications, due to the fact, that personal computers and tablet clients propably won't run the same operating system and therefore additional effort would be required to port the software. In addition, Java is preferable due to it's UI features (e.g. JFC framework) and the UI will have the same look, no matter which client is used)