UCN, University College of Northern Denmark
IT-Programme
AP Degree in Computer Science
dmai0919

1.st Semester Project

Alexandru Stefan Krausz, Sebastian Labuda, Martin Benda, Simeon Plamenov Kolev

13-12-2019

Title page

UCN, University College of Northern Denmark

IT-programme

AP Degree in Computer Science

Class: dmai0919

Participants:
Alexandru Stefan Krausz
Sebastian Labuda
Martin Benda
Simeon Plamenov Kolev

Supervisor: Mogens Holm Iversen, Gianna Belle, Dimitrios Kondylis

Abstract/Resume

In this project, our group was assigned to create a system for a retail company, which will be capable of creating orders, generating invoices and keeping stocks in the warehouses. This report presents our work and part of the system we have created.

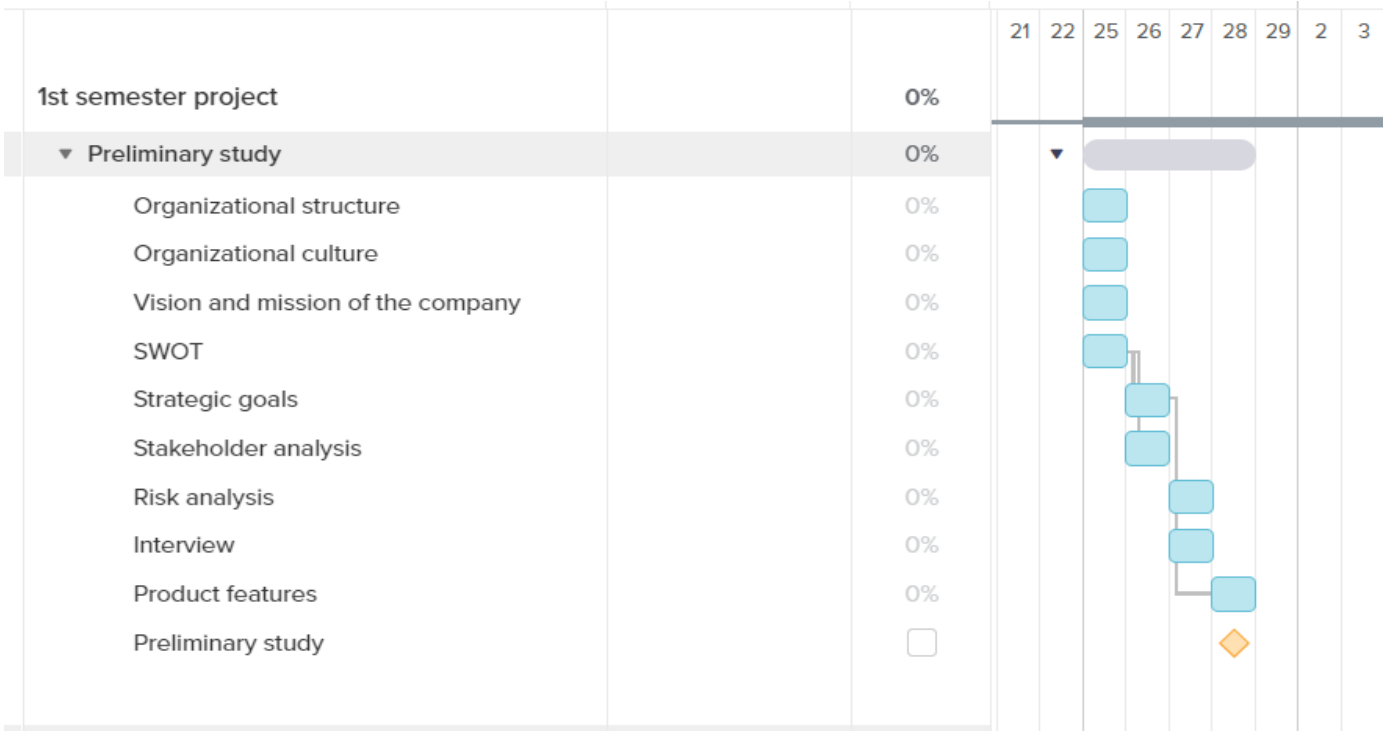
Repository path: https://kraka.ucn.dk/svn/dmai0919_1Sem_project_5
Repository number:124
Normal pages/characters: 12.58 characters /30 200 pages

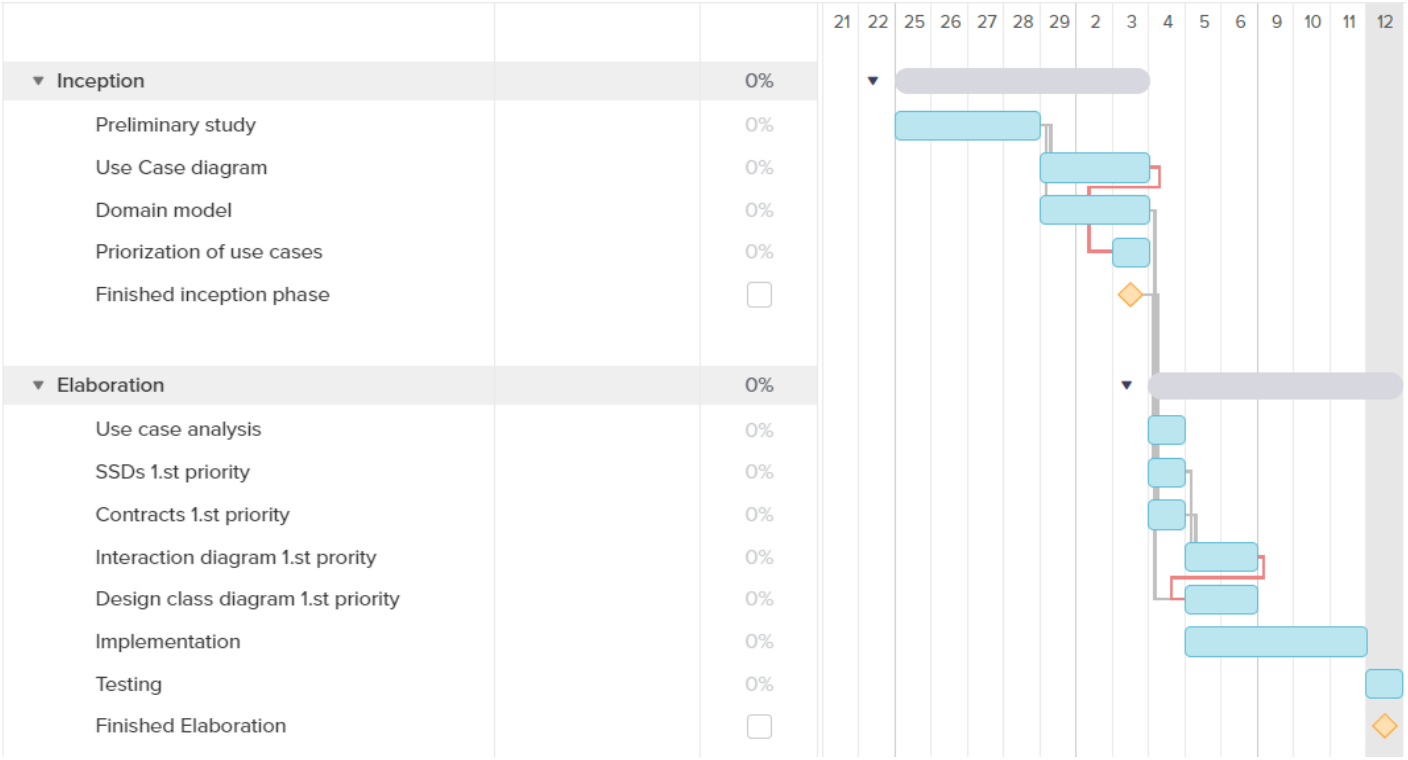
Contents

Introduction

The main reasoning behind this project, was to create a brand new system for an already existing company called Vestbjerg Byggecenter A/S. The system, that is currently used by the company is outdated in many ways. We aimed to create a program that will be easy to use (user-friendly), a system that will increase the overall effectiveness of the everyday tasks of employees, which will save both time and finances. Not only that, but the system will also include a new key feature that wasn't included within the old system. A feature that will allow employees to move products between different warehouses and be able to track their current location, see the number of products currently available and the estimated capacity of the warehouses.

Schedule





Even though we had our schedule made at the beginning, circumstances didn't allow us to follow it as intended. Already in the first week not only we had a problem with the preliminary study since we weren't sure if the things we have written were right, but also we realised that without interview there are too many things we don't understand. So instead of having most things finished until Wednesday, we couldn't do most of them until after the interview, which was at the end of that day. Most of the things after that went according to schedule plus two days we lost, but we were slowly catching up. But the moment we thought everything is alright and we were finally ahead of the schedule few problems hit us two days before the deadline and have taken their toll. Especially our inexperience and problems with the used system created big holes in our prepared schedule, which we at the end caught up.

Unified process

For the development of this system, we used the Unified process(UP). The unified process divides development into 4 phases: inception, elaboration, construction and transition. In this process developers first, analyse and implement the most complicated parts of the system. In this way, we can ensure that we will be able to create this system and in case of bigger problems, we will either solve them sooner or at least we will know where these problems are.

Inception

First, we started discussing how the company operates as a whole - who the boss is, who are the managers, what are their roles, how many employees are there and how are they distributed between the different workplaces. We prepared ourselves with a lot of questions to ask the

representative of the company who was to have an interview with our team. After receiving more detailed description of the company we managed to find out the companies organizational structure and culture, we pointed out it's strengths and weaknesses and we devised a new mission and vision for it.
Then we proceeded with analysing all the stakeholders in the company - how they contribute and how much influence they have in it. Additionally we analysed the probable risks that may befall our team during the development process and how much they would hinder us.
Next on our list of objectives is the planning of strategic goals, which would serve our team as milestones during the development, as for the company it would be the goals it strives to achieve, which would benefit it greatly both financially and in terms of time management. Together with the strategic goals, we came up with our system vision of the company, which has a really bright future for both our team and the client we are working with.
After analysing and setting our goals, we started thinking more technically and by that we began by agreeing upon the software we are going to use during development and then creating the first drafts of the workflow diagram, use cases and prioritization, domain model and a list of the product features.

Preliminary Investigation

Organizational structure and culture

Overall hierarchy of the whole company consists of three most important members. One of them is the owner and at the same time, the creator of the company Anders Olesen. Anders spends most of his time within the office while walking around motivating the rest of employees. Anders has passed most of his responsibility to his sons, managers Thomas and Casper Olesen.
The company is split into 2 departments: DIY department and timber department. While Casper is responsible for timber department, Thomas is responsible for DIY department. Not only that but Thomas is also chairman of staff, organising and arranging different events and courses for employees.

Anders expects employees to be hardworking and positive at the same time. They should remain friendly mainly towards their guests and customers, but also towards colleges and management.

Beside creator and sons, Vesbjerg Byggecenter A/S has 10 sales assistants (5 each department), 6 warehouse workers (3 each department), salesman, managers for both parts of DIY and Admin Office with 4 employees.

Work is structured within multiple groups. Managing director passes his orders to the managers. Managers then inform employees within their department.
That's why we have recognised this company's structure as a functional structure, which also describes culture in this company.

The reason for including an organizational culture is to understand the impact that our system will have on the company and provide possible solutions. We recognised the company's culture as a role culture.

Role Culture

Organisations with a role culture are based on rules. They are highly controlled, with everyone in the organisation knowing what their roles and responsibilities are. Power in a role culture is determined by a person's position (role) in the organisational structure. [1]

Role cultures are built on detailed organisational structures which are typically tall (not flat) with a long chain of command. [1]

Vestbjerg ByggeCenter is currently operating on a role-based culture because each employee fulfils a certain position(role) for example sales employee's job is to sell products both from online/phone orders and in-store. The warehouse employees add new products into the system and prepare orders for delivery. The salesman goes out on trade visits to contact construction companies with which he can then collaborate. The deputy manager is given the task to find an IT company that will be able to meet the expectations of the companies' boss. All aforementioned roles are under the managers in the hierarchy. The managers of the different departments decide on prices and discounts, create statistics, resolve disputes between employees and can replace an employee if he is absent. The role of the boss is overlooking the whole company and financing it.

A consequence of this culture and structure is that decision-making can often be painfully-slow and the organisation is less likely to take risks. In short, organisations with role cultures tend to be very bureaucratic. Right now, the company doesn't have a lot of members, so this culture doesn't affect it a lot. [1]

The new software solution that we are working on would allow the company to rapidly extend over a short period of time.
The fast growth of the company in the foreseeable future will require more employees which would make any decision making a slow and painful process because of the tall structured chain of command.

Leaders are very nice to other employees. They are trying to create a good atmosphere by helping their employees to feel better at work. This surely causes huge impact on them and motivate them within their tasks. Especially while communicating with the customers, this can help spread a positive and friendly atmosphere within the company thus increase the customer experience.

By any means, organizational structure and role culture of this company is probably the best the retailer can use. Especially this structure is used by most retailers no matter how big or small, but

the company has to be aware of the long span of control and harder communication between departments.

SWOT analysis

Point of SWOT analysis is to recognise companies strengths, weaknesses, threats and opportunities for the future. It is used to analyse what should the company improve and maintain. The company wants to compete with the biggest players in the future, which means they especially need a better online presence and functional system, which will allow the employees to work effectively.

Strengths

- Provides high-quality timber and DIY products for the consumer public
- Good mood at the workplaces
- Working with a lot of partner companies
- Functional website working with several sellers
- A senior company with many years of experience

Weaknesses

- No Twitter, Facebook profiles
- The website doesn't have all functionalities integrated
- Old IT-system

Opportunities

- Create a new up to date IT-system
- Advertise itself on SoMe
- Fully-develop the website
- Expand to the northern part

Threats

- Bigger companies (Silvan, Bilka…)

Vision and mission

Mission

The mission of the company states the main strategic goals of the company as well as its nature. Company in the current situation cannot compete against a bigger company. Instead of fighting them, the mission of the company is to expand its shops and overall awareness to the northern part of Jutland (Northern Jutland). Not only that but the secondary mission of the company is to create new partnerships within smaller companies.

Vision

Vision, as well as the expected future vision of the company, can be described as High-quality life for affordable price!

Stakeholder analysis

Stakeholders	Contribution	Interest	Influence/Attitude	Importance
Anders Olesen	Knowledge/feedback/decisions/finances	Better company	High/Positive	High
Casper Olesen	Knowledge/feedback/decisions	Better company/ easier job	High/Positive	High
Thomas Olesen	Knowledge/feedback/decisions	Better company/ easier job	High/Positive	High
Misfits	New IT-System	Satisfied customer/money	High/Positive	High
Management	Knowledge/feedback	Better company/ easier job	Medium/Positive	Medium
Employees	Knowledge/feedback	Easier job	Low/Positive	Medium
Customers	Feedback	A better experience	Low/Positive	Low
XL-byg chain	Website providers	Popularity	Low/Neutral	Low
Suppliers	Supplies	Easier supplying and orders	Low/Positive	Low

Stakeholder table is unmistakably very important part of every single project. The main reason behind Stakeholder table is to identify certain individuals or groups of people that will be involved or will in some sort of way affect current project. Stakeholder table ensures to group these organizational support depending on its level of participation, interest and overall influence in the project. The project will be well structured as well as including this main organizational players may assure their future support.

By creating an early stakeholder table, we may reduce the amount of conflicts within project and to assure key reason for organizational players being included along the project.

Table will help us to determine how best to involve each of these stakeholders and how to communicate within them throughout the project.

Risk analysis

Risk	Probability	Impact	Priority	Solution
Resources are inexperienced	3	3	9	Discuss needed features at least one day before so team can research at home if necessary
IT Staff sickness	5	2	10	Take sickness into consideration while creating plan and try to overcome it
Failure of used system	3	4	12	Always create a copy before uploading anything
Requirement is not precise	2	3	6	Try to get the most out of the interview

Risk analysis is used to analyse what can slow down the project and find a solution as fast as possible. Since already at the beginning of the project half of the team was sick, we assigned the probability to it. Despite the sickness, we are capable of working so the impact should not be that big. Since we don't have that much experience as developers, we assumed that sometimes will not be sure what to do. The system we are using and especially SVN created already some problems, so we recognised it as an important risk we should be aware of. Since everybody understands things differently, we knew that we may misunderstand or interpret differently what is required from the system, that is why recognise it as a risk. After recognising risks we created the best solutions we could to hinder those risks. Even though we thought that we were prepared enough, we didn't expected that two of the risks will hit us two days before the deadline. We expected a lot, but we didn't expect to spend at least 6 hours finding an error which BlueJ was pointing out, even though it wasn't even there. Anyway, we managed to overcome it and learn from it a lot.

Strategic goals

Plan	***Goals/Objectives***	***Relationship to project***
2019/2020 Managers strategic plan for inventory management system	Update the current system of the company	This project will ensure higher effectivity of the overall system. What is more, the system will enable to control location of products situated in warehouses.
2020 Managers strategic plan for company management system	Expand overall awareness about company within northern part of Jutland (Northern Jutland)	Since current situation doesn't allow company to compete against bigger companies, this system will ensure expanding of the company (new relationships, social media, etc.)

After we successfully managed to consider all risks alongside our project, we have move to next part towards business case, so called strategic goals. During creation of strategic goals table, we have to consider several main factors. Between most important elements belong SWOT analyzing, understanding of mission and vision of the company, as well as considering it's most harmful weaknesses. Strategic goals certainly help towards progression of the project as well as reducing possible problems during the whole process.

Product features

Feature	Description	Priority
K1	The system will reconcile with partner companies	High
K2	The system will create, read, update and delete information about different products	Medium
K3	The system will create, read, update and delete sets	Medium
K4	The system will differentiate between users with different status	Medium
K5	The system will create, read, update and delete orders	High
K6	The system will create, read, update and delete customers/employees	Medium
K7	The system will create, read, update and delete warehouses	High
K8	The system will create, read, update and delete different groups	Medium
K9	The system will generate reports and statistics	Low
K10	The system will create invoices	Medium
K11	The system shall be user friendly	High
K12	The system shall be error preventive	High
K13	The system shall be fast and responsive	High
K14	The system shall be easy to remember	High

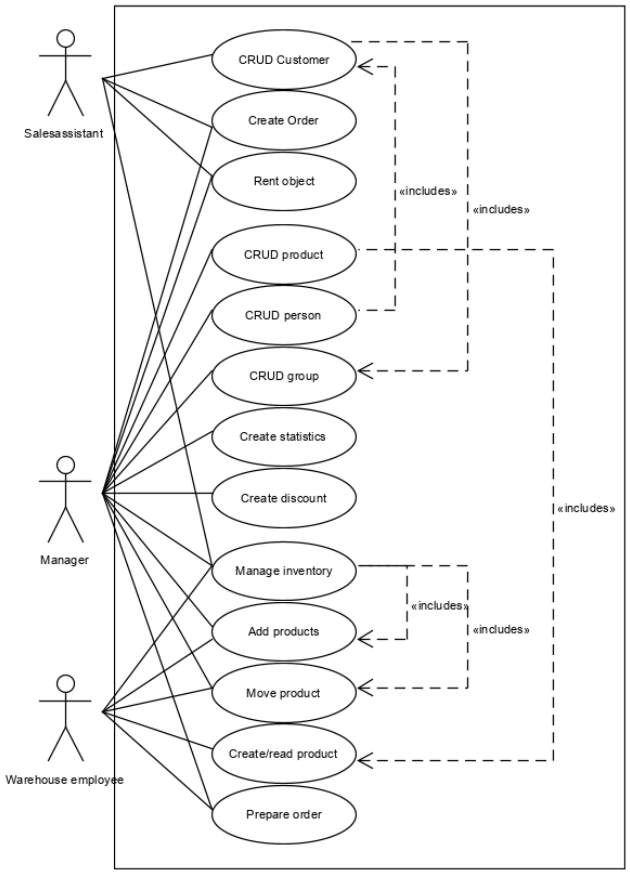
Technology

We used UMLet to create diagrams, BlueJ for the coding part and TortoiseSVN for version control.

System vision

The purpose of the project is to help a company overcome some critical problems that it has with its current system. The current one is functional but has many flaws, which makes work for employees very redundant. Old system was vulnerable due to huge amount of manual work. Our system will ensure that working will become more efficient as well as less time consuming. The product we strive to deliver will be as user-friendly as possible, easy to understand, have many new features and perform overall much better than before.

Use Cases



Use case: Manage inventory
If there is a need to add item to the stock employee will access the system, choose the warehouse, scan the bar code of the product, which stocks needs to be changed, and adds amount

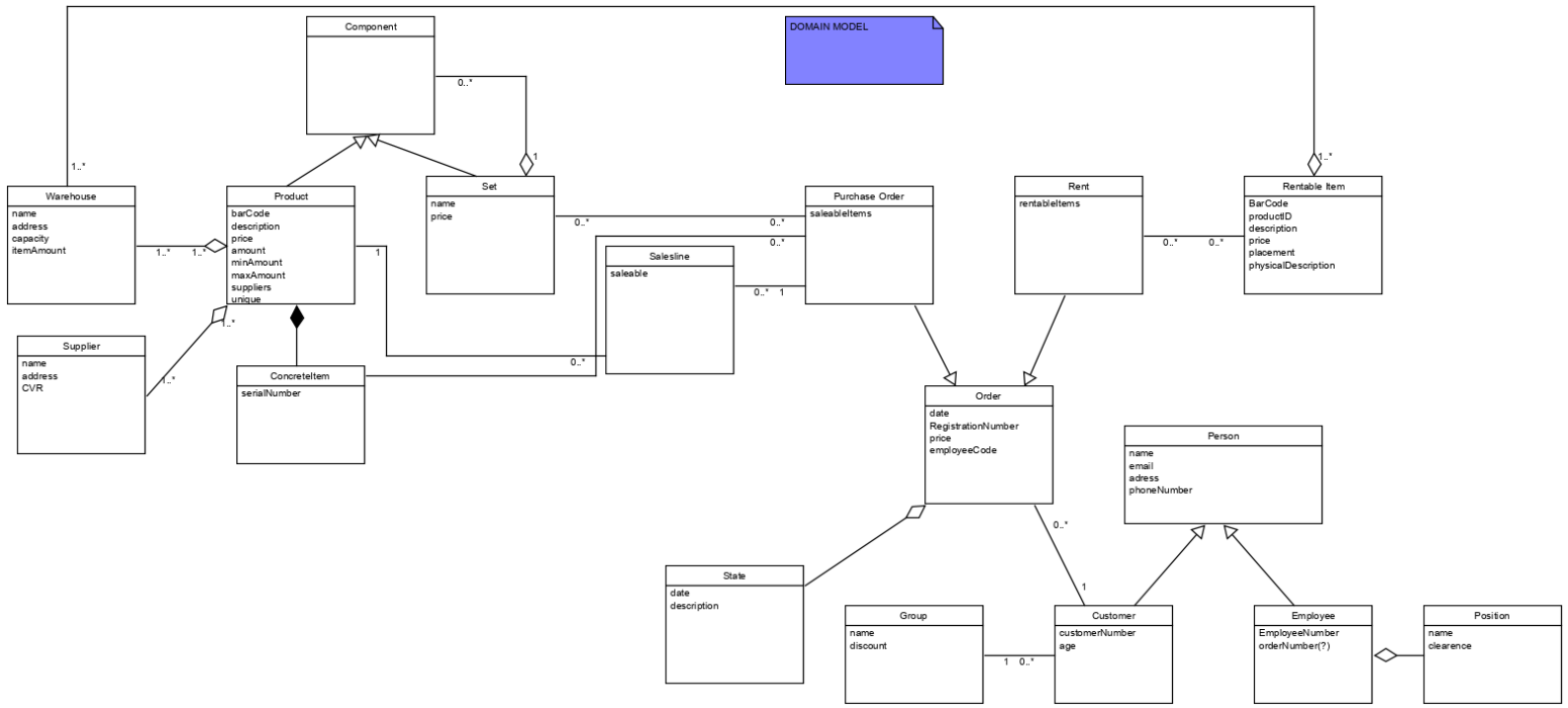
Use case: Rent object
Salesassistant or manager adds rentable items to the order, order is prepared by warehouse employee and then delivered.

Use case: Create order
Salesassistant or manager creates an order, assign a customer the order and puts items in the order. Customer either picks it up or have it delivered. In case of delivering, warehouse eployee packs the order and sends it.

Use case: Create statistics
Manager access the system and creates statistics by using

Use cases in general describe which users interact with the system and how they interact with it. In our case, we have three users: sales assistant, manager and warehouse employee. As was requested, for now, the customer doesn't interact with the system. The manager is capable of doing everything within the system. Warehouse employee can only create and read product but isn't able to delete it or update. Even though use cases "Add products" and "Move product" are part of the "Manage inventory" use case, they are more complicated, so we decided to split them. Salesassistant can only CRUD(create, read, update, delete) new customers, but he shouldn't be able to add new employees. To show this, we split these use cases.

Domain Model



The domain model was undeniably the most time consuming diagram to make because we had to change it many times due to complications. At first we had many ideas in our heads, but the problem was that we weren't exactly sure how to depict them in UML, because this is the first time we are encountering such a big project. The later versions of the domain model were much more complex, but alas we managed to put our ideas on the diagram. The problems that arose after that, were that it was actually too complex for us to implement, which is why we modified it numerous times again and at the end we had over 10 versions of the domain model.
A picture of the first draft of domain model can be found in Appendix 1.

Use case priority

	Business value	Architectural importance	Total
Manage inventory	8	8	64
Create order	3	7	21
Rent object	6	3	18
Create statistics	4	2	8

This table shows which part of the system is most complicated and most important of the project. The point of the Unified process is to first implement the first priority use case because once you can do that you should be able to implement rest of the system. Besides CRUDs, we have 4 use cases as seen in the table. Create statistics is important for the company, but it's just reading existing date so we assigned it lower architectural importance. The company doesn't have a system for renting object (at least not for keeping track of this object), that's why we gave it bigger business value, but it won't be that important and hard to implement in our system. Create order is important and it will be a huge part of our system, but the company can at this point create orders, so the business value for this part isn't that big. But manage inventory takes part in every other use case, the company doesn't have such a system right now and it is the main reason why they require a new system, that's why we assigned it the highest value.

Elaboration

Now we know what our system requires in general. Now we are able to go deep into the program and implement it and in this phase, we do the first priority use case. As described above, the point of it is to see if we are able to implement the whole system and do the most important at the beginning. In this phase, we have done diagrams which deeper analyse our program and implemented this part. Since we assigned "Manage inventory" to have the highest priority, in this phase we created diagrams and implemented exactly this use case.

Fully dressed use cases

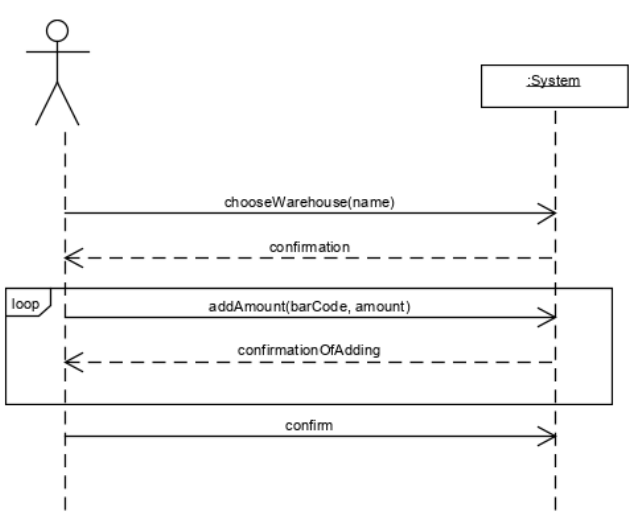
Use case name: Add products	
Actors: Manager, Warehouse employee	
Pre-conditions: employees and items exist in the system	
Post-conditions: products received and added	
Main Success Scenario:	
Actor(Action)	System(Response)
1. Employee chooses to which warehouse will be the products added.	2.System confirms finding the warehouse.
3. Employee selects products which need to be added to the warehouse an chooses amount.	4. System adds amount of the specific product to the choosen warehouse and returns confirmation.
5. Employee confirms, that he finished adding of products.	
Alternative flows: 2a: System does not find the warehouse 1. System returns fail and asks to search for the warehouse again. 4a: System does not find the product. 1. System returns fail and asks for the bar code again 4b: Maximal amount of product in stock exceeded. 1. System returns "Max ammount exceeded" message 2.Actor either chooses different bar code or confirms adding.	

Use case name: Move product(send)	
Actors: Manager, Warehouse employee	
Pre-conditions: employees and items exist in the system, there is enough space in warehouse	
Post-conditions: products sended and stored in the system	
Main Success Scenario:	
Actor(Action)	System(Response)
1. Employee chooses from which and to which warehouse will be the products moved.	2.System confirms finding of warehouses.
3. Employee selects products which need to be moved from the warehouse an chooses amount.	4. System adds the product to the list.
5. Employee confirms, that he finished moving of products.	5.System saves the list of products.
Alternative flows: 2a: System does not find the warehouse 1. System returns fail and asks to search for the warehouse again. 4a: System does not find the product. 1. System returns fail and asks for the bar code again 4b: Maximal amount of product in stock exceeded. 1. System returns "Max ammount exceeded" message 2.Actor either chooses different bar code or confirms adding.	

Use case name: Move product(receive)	
Actors: Manager, Warehouse employee	
Pre-conditions: products to receive are in the system, employee exists in the system, there is enough space in warehouse	
Post-conditions: products received and added	
Main Success Scenario:	
Actor(Action)	System(Response)
1. Employee acceses the system and chooses, which warehouse will receive the product.	2.System confirms finding of warehouse and returns lists, which can be accepted by the choosen warehouse.
3. Employee selects list of products which needs to be accepted by the warehouse.	4. System adds products to the warehouse.
Alternative flows: 2a: System does not find the warehouse 1. System returns fail and asks to search for the warehouse again. 2b: There are no lists to accept. 1. System will write message that this warehouse has nothing to accept.	

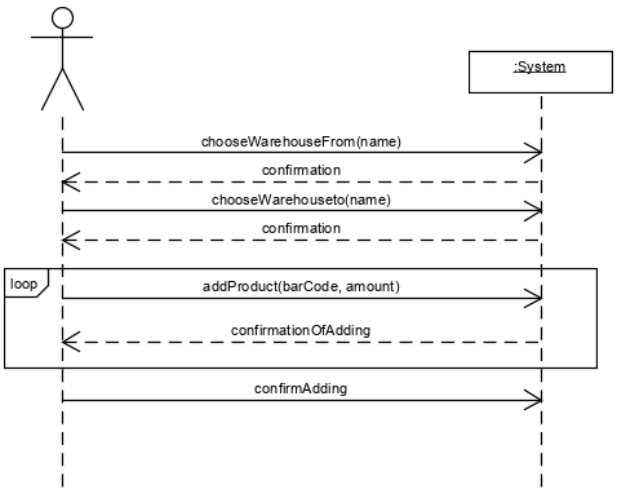
As mentioned before, add product and move product are more complicated parts of the use case “Manage inventory”, so we created separate fully dressed use cases for them. Since while moving products you need to send them from one warehouse and later accept them in the other one, we needed to separate them as well. But “Manage inventory” is at the end of the day something like CRUD warehouse, so we figured out that describing it in general in diagrams won’t be necessary.

SSDs and Contracts



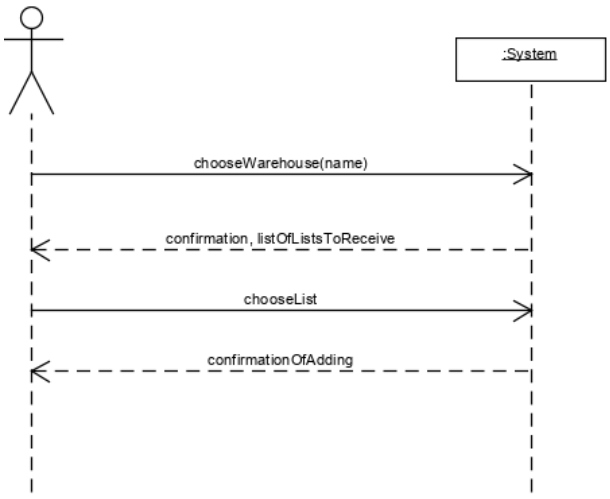
SSD & CONTRACTS
ADD PRODUCT

Contract:
Cross Reference: Add products Operation: addAmount Precondition: Products for adding exist and have been found and the amount allows their adding Warehouse exists and has been found in the system Postcondition: Products were added to the chosen warehouse



SSD & CONTRACTS
MOVE PRODUCT(SEND)

Contract:
Cross Reference: Move product Operation: confirmAdding Precondition: Products for sending exist, have been found and assigned to the list. Warehouses exists and has been found in the system Postcondition: List with assigned products was added to the TransferList.

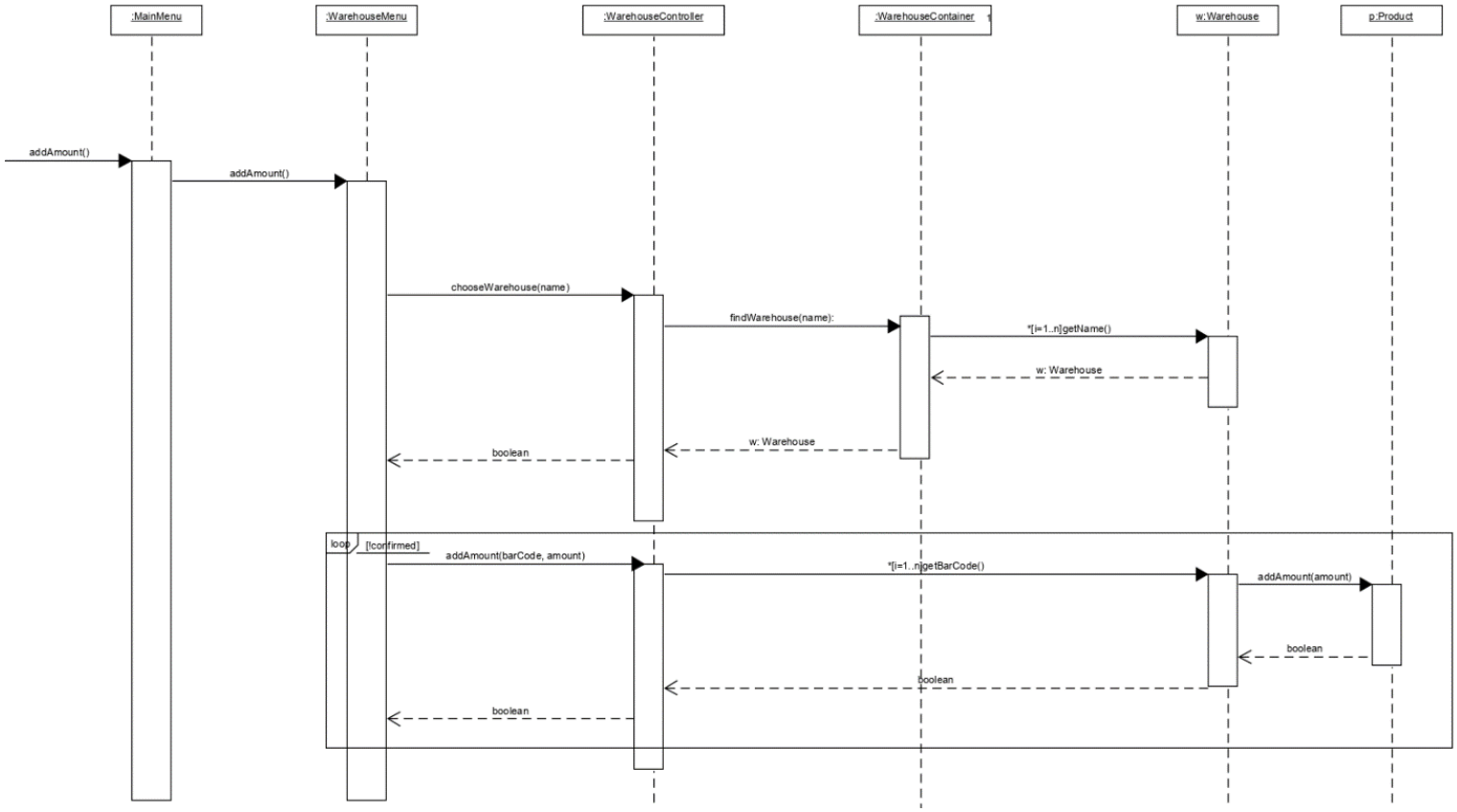


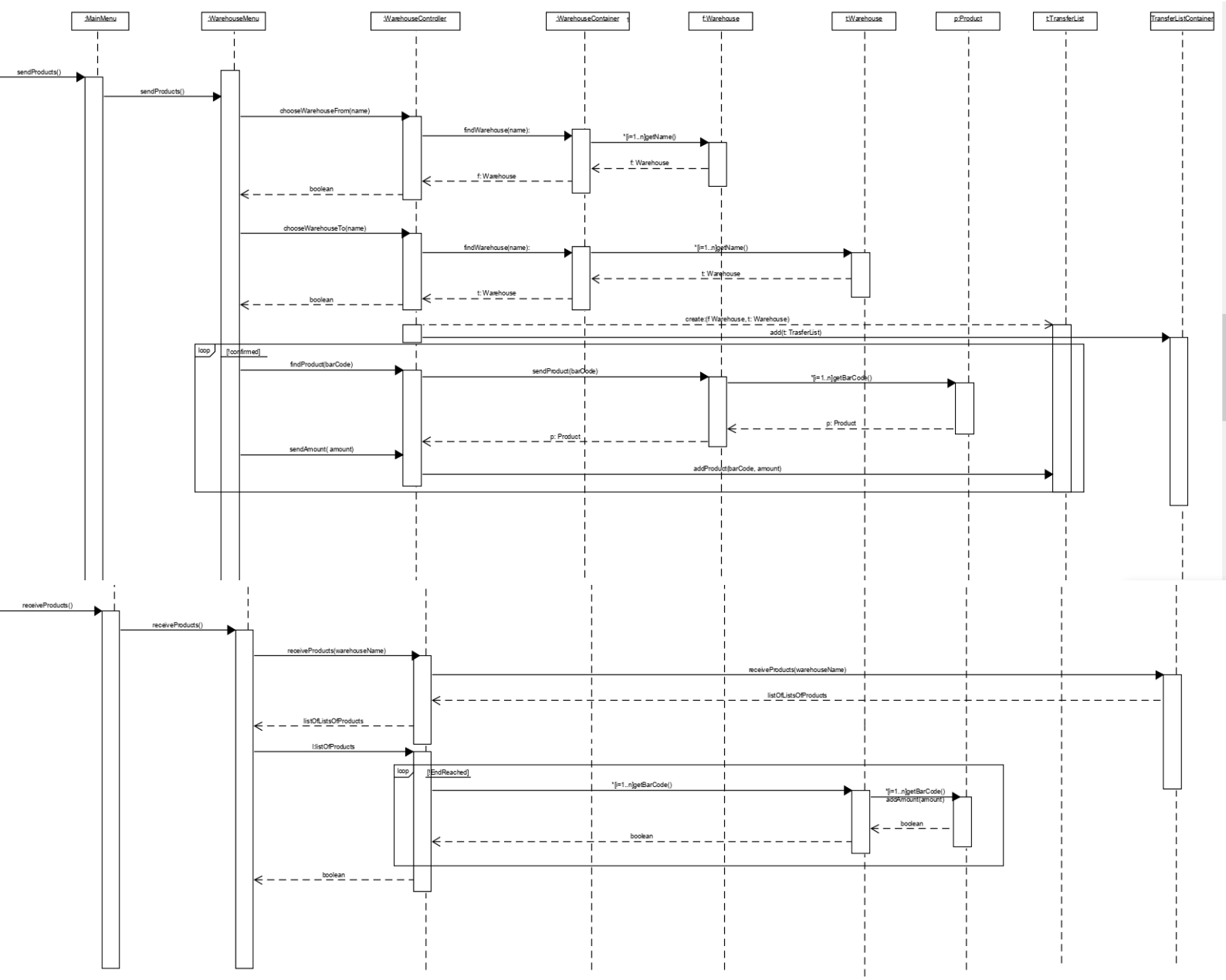
SSD & CONTRACTS
MOVE PRODUCT(RECEIVE)

Contract:
Cross Reference: Mange inventory Operation: chooseList Precondition: Products for receiving exist and have been found and the amo allows their adding Warehouse exists and has been found in the system Postcondition: Products were received into the selected warehouse

System sequence diagrams (seen on the left part of the pictures) show how the user interacts with the system and how the system responds to his action. Those are created from fully dressed use cases. In the first one, user chooses, which warehouse should receive the products and then he scans the barcode and add the amount of how much should be received. While sending products, a user first chooses which warehouse is sending products, which one should receive them and adds items to the list of products which will be sent. While receiving products from another warehouse, the user receives lists of products, which specific warehouse should receive and chooses, which of these lists will they receive from these and our domain model, we created contracts (seen on the right). The contracts are supposed to help us to organize and have a better overlook of system changes when they happen. Contracts explicitly show when is something changed in the system, like in our case when products are added, a list of products is created and stored or a list is received.

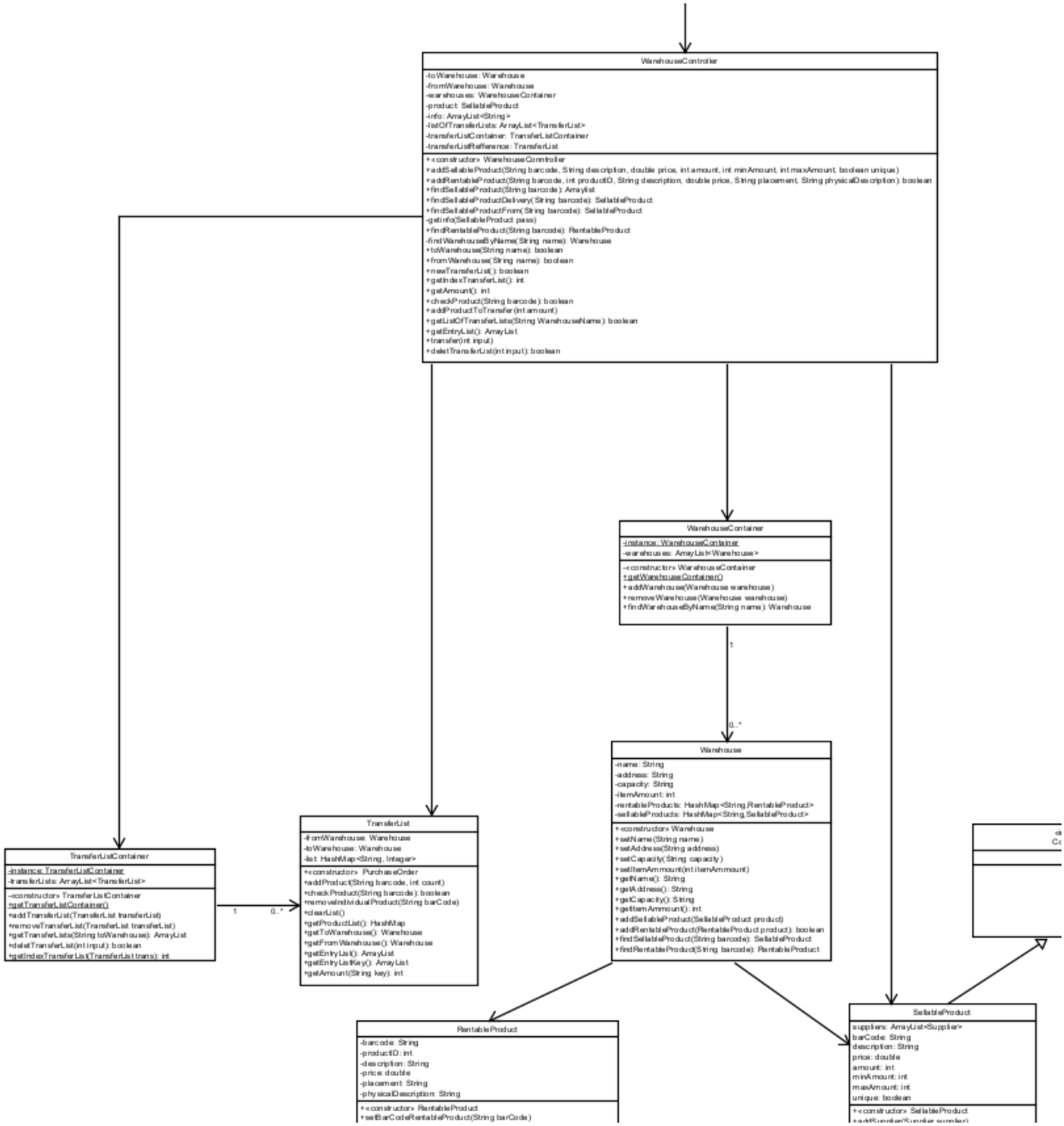
Interaction diagram





For our interaction diagram, we have decided to use the sequence diagram since it's easy to read and to work with. The problem with these diagrams is that, as seen in the pictures, it's too big, so we added it to our repository if needed. This diagrams describe how are the information passed, who creates which object and who calls methods. In these diagrams, you can see how adding products, sending products to another warehouse and receive these products works. In all cases you first need to choose warehouse, which will accept/send products. Then for example while sending products to another warehouse, you need to choose also to which warehouse will you send the products. After that, the list is created and added to the container. User adds items and amount to this list and at the end, confirms that he finished adding, as can bee seen on the guard of the loop.

Design class diagram



Design class diagrams display all the classes, methods and fields in the system, return types and which classes access which. Since its really big we included it in the repository path, the whole one is in the appendix as well. In this picture, you can see the most important part of our system at this point, the one important for our 1.st priority use case. At the top is the warehouseController, which accesses warehouse container(holding all the warehouses), sellableProducts, TransferList and TransferListContainer. While sending products from one warehouse to another, list of products, which need to be sent, is created in TransferList and then stored in the TransferListContainer. To accept them, WarehouseController passes to the UI all the lists in the TransferListContainer, which should be accepted by the specific warehouse. While accepting

products, Controller accesses the warehouse, in which the number of specific products needs to be changed.

Architecture

The architecture type that we used for our software solution is a closed 3 layer architecture.

A 3-tier architecture is a type of software architecture which is composed of three “tiers” or “layers” of logical computing. They are often used in applications as a specific type of client-server system. 3-tier architectures provide many benefits for production and development environments by modularizing the user interface, business logic, and data storage layers. Doing so gives greater flexibility to development teams by allowing them to update a specific part of an application independently of the other parts.[3]

TUI(Presentation layer) -> Controller(Application layer) -> Model(Data layer)

It is a closed architecture because the TUI(Presentation layer) doesn’t read the Model(Data layer) by himself but only acts as an interaction interface between the user and the system.

User Manual

For moving products between warehouses use [1]Manage inventory.

In order to send products use[1]Move product:

-Search the warehouses by name(From and To) enter ”Exit” to exit at any time during this action

-Search the products by barcode(manual input or scan) enter ”Exit” to exit at any time during this action

-Enter amount to be transferred, enter ”0” not to add the specific product

-Enter “Done” when you added all products that need to be moved.

In order to receive products that have to be delivered to your warehouse use [2]Receive delivery:

-Search for your warehouse enter ”Exit” to exit at any time during this action

-Choose the number corresponding to the list you are receiving enter ”-1” to exit at any time during this action

In order to add a new product in the system use [3]Add new product:

-Search warehouse by name enter ”Exit” to exit at any time during this action

-Enter product information enter ”Exit” for names and barcode or “-1” for amounts to exit at any time during this action

In order to search for a product use [4]Search product:

-Search warehouse by name enter ”Exit” to exit at any time during this action

-Search product by barcode(manual input or scan) enter ”-1” to exit at any time during this action

Code Standards

Implementation of the code required us to agree upon a code standard to make the code style consistent and readable throughout the entire application. Unified code standards made the project easily maintainable, readable and extendable. Before starting to program, we agreed to use the official Java Code Conventions[2]. By following them, we properly used spacing, indentation and other good practices. Method names should state their function in the system. Additionally all methods were commented with JavaDoc.

Groupwork

Already while creating the group, the point was to have a group of members willing to improve and learn while having a similar personality capable of working together. Especially the learning part was important. Creating a balanced group where everyone likes similar things and thinks in a similar way is very hard and rare, but I think that is what we have managed. Every member of this group wants to participate and wants to do as much as possible to improve, so we don’t have any conflicts between members. Already after forming this group we exchanged the result of our Insight testing, from which we not only learnt more about other members and how to understand them but also created roles in our contract. Thanks to this we feel that we are part of something unique, which motivates us even more. We had a lot of fun making this project. During our work, we realised similarities between this project and our future projects as professional developers. We have encountered many conflicts, such as uncertainty of things we are doing or the way we have done them, misunderstanding of how the system should work or sudden change and complication few days before handling this report. Anyway thanks to our hard work and high participation we managed to overwhelm them.

Group Contract

We in the group 5 called Misfits solemnly swear that we are going to do our best work to achieve our goals. The whole group is going to assign individual work according to our knowledge and possibly according to our preferences.

As a group, we will try to create the best product we are able to, which means members of this group will do everything to achieve that. But the goal of this group is not to do the best, but to learn as much as possible and every failure means a new lesson for us.

No member is allowed to be slacking. If someone comes late or is slacking without a good reason or is purposely not helping with the group work, he shall be named traitor and will buy one

desired item (which price may not exceed the average price of beer in a bar) for every other member. The whole group decides if the member's excuse is good enough or not.

Roles

Every member of group Misfits is assigned a role based on his personality. The role doesn't have to be obligatorily followed, it is just to remind all members what fits them most and where they can be most efficient.

Alexandru Krausz: Carry

Martin Benda: Carry

Simeon Plamenov Kolev: Tank

Sebastián Labuda: Support

Carry:

Carry's job is to work on the most complex parts of the project. They focus on their work to ensure that the final product of the group will be functional and useable.

Tank:

Tank should create a basic structure of the project. Catching errors and misses, they provide stability to the group's work, thus protect it. They also have to work on more complex parts, especially to have a better overview to protect the project as a whole.

Support:

Support provide overall help to every other member. They keep track of other members and coordinate the team, hence they have the best overview of the work done. Professional or personal help, they need to ensure that the group is in the best shape.

This contract bounds Martin Benda, Simeon Plamenov Kolev, Sebastián Labuda and Alexandru Krausz and shall not be broken until group members depart their ways.

Motto: If there are no problems we will make them.

Conclusion

In conclusion, we can say that this project was a great experience for us. Even though we didn't manage to do as much as we wanted and our inexperience and small carelessness nearly destroyed all our efforts, we enjoyed it. Thanks to our fails we could learn from this project far more than without them. Not only it helped us to better understand the business part of the software development, but also how much should risks be considered, the importance of the well-done schedule and last but not least proved the importance of diagrams. It showed us how exactly the whole developing process looks like and through practice prepared us for our future even more.

Literature

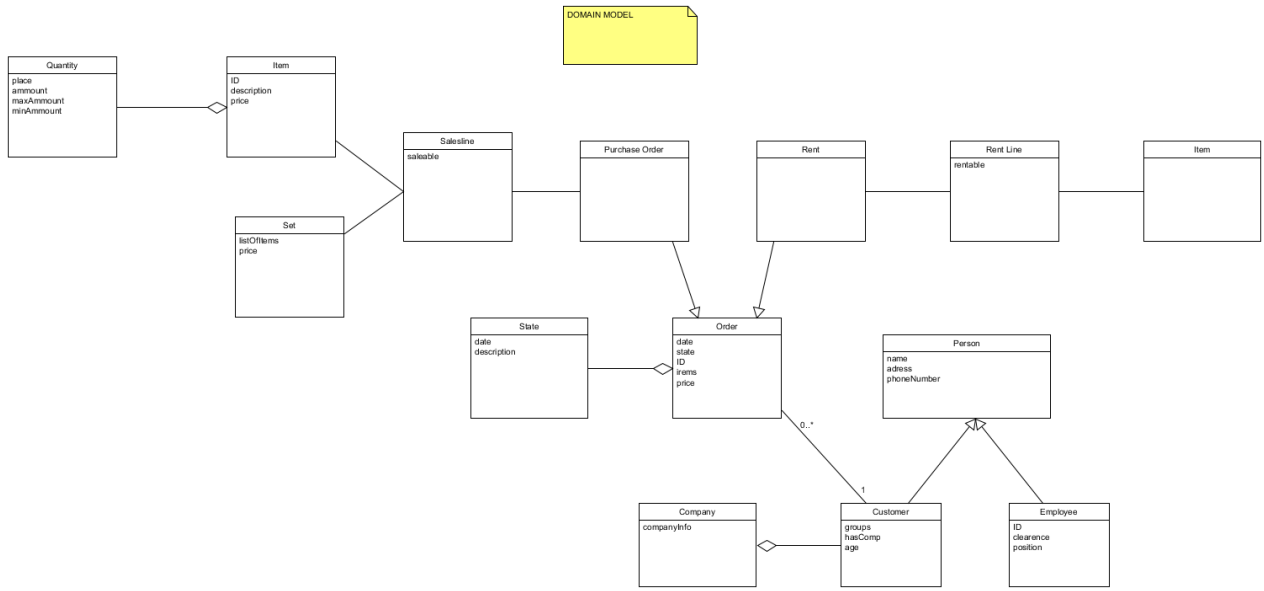
[1] Jim Riley, “Handy’s Model of Organisational Culture | tutor2u Business,” 2019. [Online]. Available: https://www.tutor2u.net/business/reference/models-of-organisational-culture-handy. [Accessed: 25-Nov-2019].

[2] Oracle.com, 2019. [Online]. Available: https://www.oracle.com/technetwork/java/codeconventions-150003.pdf. [Accessed: 25-Nov-2019].

[3] [1] JReport, “3-Tier Architecture: A Complete Overview,” 2018. [Online]. Available: https://www.jinfonet.com/resources/bi-defined/3-tier-architecture-complete-overview/. [Accessed: 13-Dec-2019].

Appendices

Appendix 1 - First Draft of Domain Model



Appendix 2 - Design class diagram