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

2nd Semester Project

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

29-05-2020

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: Gianna Belle

Abstract/Resume

In this project, our group has been assigned to create a brand new system for four year old company specialized in producing different types of machines. Since the company for which we are working doesn't have any kind of system, we will be creating a brand new system, which will be capable of solving most of the problems within working area. Between the most important features belongs to the ability to receive offers, create orders and scheduling work within the company. Thanks to this project, we will be able to strengthen our knowledge and experience and to create more advanced type of system.

Repository path: https://kraka.ucn.dk/svn/dmai0919_2Sem_4/

Repository number: 175

Database name: dmai0919_1081492

Normal pages/characters: 89330 characters / 37.22 pages

Contents

Introduction

It is always a pleasure for university students to get any kind opportunity, through which they can practice the knowledge they have gained during days of studying. Second semester was still special due to the type of task we received for the 2nd semester project. The task consisted of finding a company and creating a new system depending on the selected company. We have received great chances to work for companies included within the machinery industry market. The name of the company is “Ms Stahl Milling Srl”. The company specializes in producing precision milled components mostly for other companies and currently consist of two members. Nonetheless, the

company is becoming very popular mainly for its affordable prices and high quality products. With this being said, there is a problem rising from the horizon. Company doesn't have any kind of system whatsoever and the owner of the company is well aware of this problem.

Problem area

Since the company doesn't have any kind of system, all the documentation is done through paperwork, which costs the company both a lot of time as well as finances. Newly developed system would definitely boost overall production and efficiency of the company. Not only that, but the company will thanks to the system have much better control over its employees and all orders.

Anyhow the company is currently standing in a competitive area. Current market consists of other rivals who are already way more experienced and have way bigger awareness over the country. What is more, the owner of the company has been planning to expand his company for quite a long time, yet he is not experienced in this matter. Expanding can be both very beneficial or sometimes lethal. The last problem can be considered the fact that the company has currently no advertisement. At the end, the company has a limited amount of finances and for that, we need to approach this task with maximal caution.

Problem statement

The issue outlined above leads to the following problem statement

- How should the company proceed in terms of expanding while being in the current market ?
- What type of resources should the company use to ensure successful advertisement ?
- How can a company be sure that investing into a new system will prove beneficial ?

Method and theory

One of the first steps included collecting empirical material in the form of interviews with the owner of the company. Another important information consisted of the mission and vision of the company or even its previous strategies. After collecting all necessary data, we have reconstructed the current situation and came up with results.

As for the first answer, patience will be the key to victory. Surely, the market is full of different rivals. Company can however make connections with bigger companies. This way the company will gain a lot of new customers as well as new contracts. As for expanding, the owner of the company will have to find balance between buying new machines, hiring new employees and still remaining financially stable. Spreading of a company is a long run process, the market is changing quite often, so the strategy should be always updated up to date.

For the second question, there are of course multiple solutions. However we would recommend companies to use any kind of social media (for example facebook). The reasoning behind this fact is quite simple. Creating a company on facebook doesn't require any payment, news and information are quickly spread within social media, so the company will have a high chance of gaining new customers very quickly.

For the very last question, at some point it has been answered from a certain point of view. The way the company works with paperworks is truly outdated. Employees have to stay for long hours to fill all the documents. This way, a company has to pay much more to its employees and wastes a lot of precious time, which can be spent much more effectively. Company can pay much more to its customer base, accepting more offers or even gaining very valuable resources within their market.

Unified Process

There are multiple software development processes that developers can use in order to deliver their final product, each of them has different pros and cons. Some are very strict and can be very risky when not defined properly right at the beginning, others are so-called “agile” and allow fast changes and adaptation. For this project, we decided to use the Unified Process.

Unified Process is iterative, meaning it divides the whole project into small iterations with defined requirements, design, implementation and tests. There are multiple benefits of the iterative approach. It is possible to get feedback after each iteration, so the probability of finding misunderstandings with the stakeholders is much bigger. This can ensure customers satisfaction as well as adaptability to sudden changes.

Another characteristic of the Unified Process is being architecture driven. The architecture of the system is decided and built in the early stages. This prevents….

This process is also use-case and risk driven. Unified Process uses Use Case Model to define functionalities of the system. In order to deal with them, it divides the functionalities from the highest priority to the lowest and starts with the highest one, in order to reduce uncertainty. Since it first tackles problems with highest priorities, the risks and problems can be found in a very early stage of the development.

Unified Process divides development into 4 phases: inception, elaboration, construction and transition and 6 disciplines: business modelling, requirements, analysis and design, implementation, testing and deployment, as can be seen in Figure 1. Except for the Inception phase, all other phases are divided into iterations.Discipline Business Modeling focuses on understanding the organization, the way it works and what does the company consist of. In the Requirements discipline the system vision is defined, all the necessary parts of the system as well as who interacts with the system and how they interact with it. Analysis and Design discipline analyzes all the requirements and defines the system, such as the architecture, system interaction etc. . Implementation is simply implementing code for the designed system as well as testing of separate units. During the Testing discipline, tests are prepared and all the interaction and system tests are executed. Last discipline of the Unified Process is Deployment, where the acceptance tests are undergoing, the system is planned and installed in the client's environment and end users are trained for the new system ,

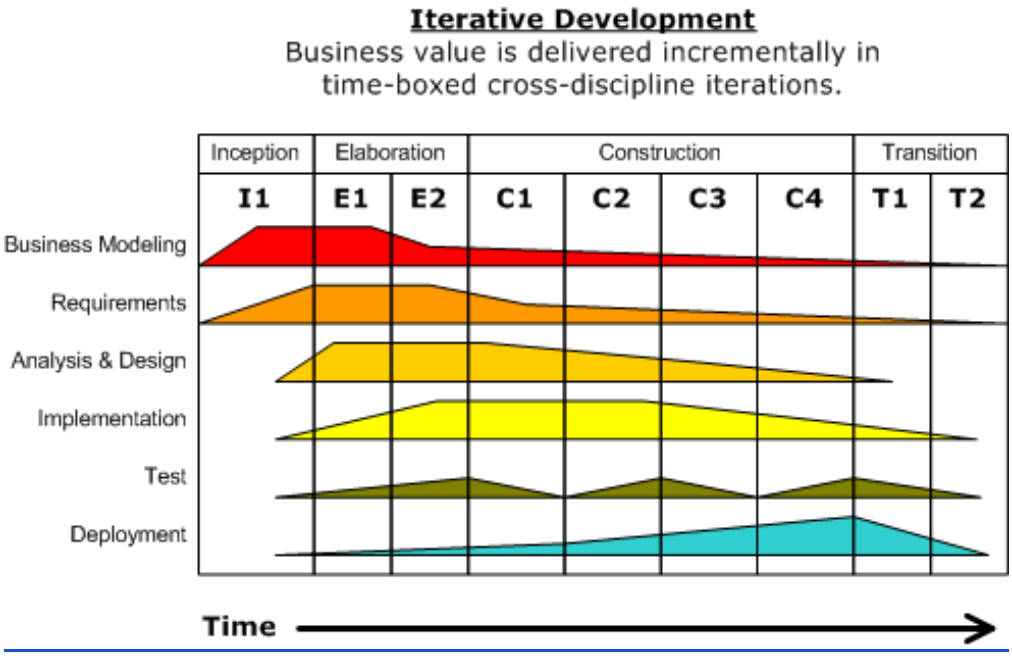


Fig.1 [1], Phases of the Unified Process

UP Plan

Important part of development of every system is creating a schedule and proper planning. Without it, it would be unclear when to finish which parts of the system, which would create confusion and put the whole project behind the schedule more than once. With bigger systems this has an even larger impact. The UP plan sets a goal for each phase of the Unified Process, its deadline, how many iterations will the phase include and which artifacts should be finished in certain phases.

Even though we created our UP plan (Fig.2) at the beginning of the project and refined it properly in the inception phase, there were many obstacles which didn't allow us to follow its deadlines. First and the biggest one was the current pandemic situations, which changed the way people work. We faced multiple new challenges, such as working from home, adapt to the current situation in the world, which was one of key elements in changing our attitude, lifestyle and overall disposition. These complications of course affected our overall scheduling. Situation required our team to adapt, consisting of changing our scheduler multiple times. We clearly realised that writing a proper schedule, which will work without or with only small changes would be impossible .

Another problem was that some things we needed to use and implement in this project were taught later or at the end of the semester, so we did not know what to expect, how complex it would be and how it would change or affect our semester project.

Phase	Inception	Elaboration	Construction	Transition
Goal	Vision and scope business case, requirement understanding	Skeleton architecture in place, most important use cases are designed, implemented and tested	Full system designed, implemented and tested	Roll out the fully functional system to customers
Number of iterations		2	4	2
Deadline	31-03-2020	30-04-2020	29-05-2020	???
Artifacts	• Business Case • Strategic alignments • Product features • Risk analysis • System vision • Mock-ups • Use cases • Plan and estimates	• Domain Model • Use cases and SSD • Interaction diagram • Database • Implementation (fully dressed use case(s)) • UI-design • Test Cases/Test	• Revisions of use cases • Use case design • Implementation • Test Cases/Test • Documentation • Preparation for deployment	• Beta tests • Deployment

Fig.2 , UP plan

Preliminary Study

Organizational structure and culture

Organizational structure and culture are a very important part of every company. The structure of the company is also often called its skeleton. It defines who is responsible to whom, how employees coordinate together and which tasks they have. The culture of the company includes many things, so it is hard to define. It consists of belief, values and principles of employees. It defines how employees behave, how they interact, how adaptive or resistant are they to changes.

Recognizing organizational structure and culture is very important for every company. Having a proper structure for certain business can dramatically increase its effectiveness and help to avoid conflicts and misunderstandings. Proper structure and increase adaptability of the company or helps to do all necessary tasks more effectively.
On the other hand, culture of company represents sets of values, goals and practises respected within selected company. The environment in which coworkers work can drastically improve their overall happiness. Happy workers are for example much more effective and productive, likely to come up with new ideas and to show more interest in the working area. Not only that but satisfied workers will more likely stay within company and accept unavoidable changes in the selected company.

The company we are working with presently consists of two employees, namely the owner and the accountant. Because of this fact, there is no culture or structure built or rather it is very insignificant and hard to define. But MS Stahl Milling became very successful and it is clear that they will require more employees in order to grow. This can mean very sudden growth and uncontrollable changes in the structure and culture of this company. Therefore it is very important to understand different structures and cultures, knowing their pros and cons and trying to set up the most appropriate one. There are several cultures and structures this company can end up with.

There are multiple structures the company could end up with, the most common ones are functional, multidivisional and matrix.
Functional structure divides organization into functional units(departments), where each of them has a separate task in the company. Because of that, each department normally has a manager responsible for the department. In this organizational structure, roles and responsibilities are clearly fixed, with people with the same skills grouped in one department. Employees can also see possibilities of career growth, which normally motivates them. The communication within the department is excellent, but communication between departments is very slow. Because of this and bureaucratic hierarchy, decision making takes a lot of time, so companies with this structure tend to be stable, but much less flexible.
Multidivisional structure divides the company into self sufficient divisions. Each division has their own management, equipment, supplies and resources, which gives them more autonomous work and quick decision making within the division. Because of this autonomy, the division can sometimes see itself as a separate company, which will make them concerned only about their problems, making cooperation between divisions much harder. Since each division can work on their own, more employees can accomplish all tasks, which increases costs.
Matrix is a project based structure, where based on the project employees can be responsible to multiple managers. Since it is a project base, human resources are used as effectively as possible since they can be swapped between project teams whenever need arises. This also helps spread information and see the bigger picture. Companies with this structure tend to be very flexible and adaptive to changes. Since there are two bosses, employees are often unsure, to whom they are responsible. Because of that, multiple conflicts can arise between managers as well. This structure is very complex and often requires more management costs.

Because MS Stahl Milling is a small company and even if rapid growth it won't become a huge company very soon, multidivisional structure is rather out of question, since it is more useful for large and often international companies. Even though work of this company could be seen as project based, it differs from the project in a way that the work cannot be done simultaneously but one after another, so it will not gain as much from the flexibility and effectiveness of matrix. That is why we think, after a bit of growth, this company would benefit the most from functional structure.

Naturally there are too many things influencing company culture to categorize it, but from a structural view we could divide it into 4 types: power culture, role culture, task culture and person culture.
Power culture is often found in small companies. All the power is focused in few individuals and their decision takes place. In this culture everything depends on these individuals and trust of the coworkers in this individuals. Although this culture can be very strong if these coworkers strongly believe in those with power, this culture can easily become toxic and with too many people it can become uncontrollable.
Role culture is based on rules. Everybody is assigned a certain role and responsibilities. This culture is mostly built on the organizational structure. Difficulty of this culture is decision making, since organizations with this culture tend to be very bureaucratic.
Task culture is often found in matrix organizations. Since matrix is a project based organization, where team members can often change depending on the project, the effectiveness of the group is determined by individuals, their personalities and leadership. Since everyone is working nearly autonomously, dividing work into separate tasks executed by individuals (specialization) can be very difficult.
Person culture is represented by individuals. There is no formal control and the organization exists only because of mutual benefit for individuals. Influence in this company is based on expertise and respect.

For MS Stahl Milling, we think that the best fit would be the role culture. Even right now, specialization is partially existing within the organization, since there needs to be someone who programs the model of the product, communicates with customers, manages finances and creates the actual product. Every individual working autonomously wouldn't be of much help for the company, since it would slow down the work and would require expertise in many fields and it is rather rare to find an individual with all those qualities. This way, task and person culture are much less effective. Power culture could work especially from the beginning and it could be a good solution for the company for a while, but later on, when the company will grow, role culture is the most fitting one for this type of business.

Porter's 5 forces

Porter's five forces is a tool for understanding competition and how competitive the company is. It is an important tool, because recognising the competition can show the company what they are lacking and where they should improve. It also presents new entrants and how easy the company can be replaced.

Buy-side threats:

- power of suppliers
- power of intermediaries

Taking into consideration the supplier part of their supply chain management, the products MS Stahl Milling makes are pretty complex, so numerous suppliers will exist at the current time, which are providing all necessary parts for the specific orders. We may not have access to the supply chain, but it is important for the company to have multiple suppliers assured and to always check for new suppliers to have an optimal supply chain and to ensure best offers.

Sell-side threats:

- customer power and knowledge
- power of intermediaries (price comparison sites)

Since this business right now is very popular and demanded and there are many existing companies with different prices, services, etc. and provided that customers tend to research prices and check reviews beforehand, the company(MS Stahl Milling) is definitely in a difficult and competitive situation.

Competitive threats:

- new entrants
- new digital products
- new business models (e.g new service delivery, ...)

When it comes to competitive rivalry, there is a vast amount of bigger and smaller companies providing the same services. Some of them have exponentially good reviews and have a well-established customer base.

The company is under a big threat of substitution. Since everybody needs their services, many already existing companies are delivering similar or same products. There is a probability of new entrants, but chances are lower since the complexity of creating these products is high

Although the company is very successful, they need everything they can in order to preserve it. Because of the huge competition in this field, constant investments in improvements will be necessary to ensure competitiveness.

SWOT and TOWS analysis

In comparison to Porter's five forces, SWOT briefly analyzes also positive sides of the company, not only threads. It is used to find out strengths of the company and to build upon them, weaknesses of the company in order to improve them, opportunities which could improve the company and threads which can endanger it. While strengths and weaknesses are from the internal environment, opportunities and threats are from the external environment. TOWS analysis is a variant of SWOT analysis, where external and internal factors are combined together to make conclusions and to build strategy for the organization.

SWOT Analysis

Strengths:

- Competitive prices
- Connections with bigger companies

Weaknesses:

- Small company
- Little to none advertisement

Opportunities:

- Opportunity for quick expansion
- Connection to bigger companies resulting in high-value contracts and long term collaboration

Threats:

- There is a lot of competition for contracts in this business area.
- Too quick expansion resulting in failure.

TOWS Analysis

Strength - Opportunities:

- Competitive prices, resulting in high success and earnings and opportunity to expand.
- Connections with bigger companies, resulting in a gain of popularity, collaborations and opportunity to work with even more companies.

Strength - Threats:

- Use the connections with the bigger companies to boost the number of contracts received in this business area.

Weaknesses - Opportunities:

- Use connections to advertise and gain popularity.
- Expand further to become a bigger company.

Weaknesses - Threats:

- Keep the company in a manageable size.
- Focus on social media and already existing connections to gain popularity.

As we can clearly see, both SWOT and TOWS serves as valuable tool analysing for both external and internal environment of the selected company. Recognizing companies weaknesses as well as its threats serves as great start into countering them with robust amount of whole scale of different

opportunities and its strengths. Thanks to these analysis, the company should be able to clearly set basic strategy plans and to make efficient decision, if an important or difficult situation should occur. It is thanks to SWOT simplicity, yet comprehensive way of assessing both positive and negative attributes within and without the company. The biggest opportunities as well as strengths for our company became mostly its connection to bigger/more progressed companies and competitive prices. It is mainly because of collaboration with other companies and its prices to increase popularity and overall awareness for the company. On the other hand, the highest risk for company consists of its current size, which may end up in both reducing amount of contracts or even worse, uncontrollable expansion that may end up by complete failure.

Strategy process model

Strategy management is well known among businesses as process of creating company's/organization's strategy for achieving overall long-term goals. For managers, it will surely serve as set of choices, which should enable company to achieve better performance. As we may assume, strategic planning process should include a situational analysis. Those consist of going through current internal and external environment in finds itself in. Thanks to these environmental assessments, the company can implement and evaluate basic strategic objectives. Interesting yet important fact remains that depending on markets dynamicity, the strategy process plan may cover from 2 to 5 year period.

Strategy planning process consists of four main steps:

1. Environmental scanning

   The environmental scanning idea has been described already within general description of whole strategy process model. In this part, we mostly want to use our already existing SWOT analysis, serving as examination of company's strengths, weaknesses, opportunities and threats. Another part of environmental scanning depends on providing analyzation of internal and external factors that influence selected company. These factors include rivaling companies/organizations, reconsidering the power of your suppliers, even the power of your buyers and customers. This way, we should be able to address possible threats or whether there are any obstacles before entering new market.

2. Strategy formulation

   Process of strategy formulation consists of deciding, what is best course of action for achieving all the organizational purposes as well as its objectives. From this point, the company will develop its vision and mission statement. By setting these important objectives, company will clearly describe its future plans, essential values and which direction the whole company aimes. After defining all of this values, the next step consists of formulating detailed strategy for achieving selected goals.

3. Strategy Implementation

Strategy implementation insinuates upon making the selected strategy to work properly, speaking otherwise, putting selected strategy of company into the action. This implementation starts mostly by creating a set of objectives that will lead towards selected goal/vision. There can be different sorts of objectives, but mostly we try to focus on something more specific. For example, objective can resemble from raising total revenue of company by specific percentage, attracting more customers by the end of stated time or even recruiting more employees like in by us selected company, due to its small amount.

4. Strategy evaluation

   Implementation of overall strategy, which was selected may differ depending on type/specialization of company. For comparison, bigger companies mostly implement evaluation by dividing work between set of organizational members than the members that created the plan itself. Basic meaning stands behind constant monitoring plan, alongside that assessing, if the plan is achieving or going towards stated objectives (strategic goals). Measuring performance and taking corrective actions in this stage will assure that implementation will meet the organizational objectives.

Objectives:

1. Raise the total revenue by at least 25% by the end of the year.
2. Attract more customers, increase customer base by 20% by the end of the year.
3. Buy more production equipment, having enough machines to meet the needs of new employees
4. Recruit more employees, employ at least 5 people by the end of the year.
5. Expand HQ and further in the country, have one more department in the next several years.

Strategies to achieve goals:

1. Balance between recruiting more employees, purchasing technical equipment and attracting customers.
2. Focus more on advertisements, social media and connections with bigger companies.
3. Attract more customers, resulting in a bigger revenue, which can be used for purchasing machinery, which on its own will have a big impact on total revenue.
4. Post job offers around the social media, get recommendations from bigger companies.
5. Focus on all previous goals.

Key performance indicators (critical success factors):

1. Raise the total revenue by 10% by the end of half of the year.
2. Have a customer base 10% bigger than the initial one by the end of half of the year.
3. Purchase a new machine by the end of half of the year.
4. Employ at least 2 people by the end of half of the year.
5. Expand the main department / Create a new department.

After multiple conversations with owner of the company, as well as considering all of its weaknesses and strengths, we manage to came up with final version of our strategy process model. Our group had to consider multiple factors, as we are speaking about company, which is still quiet new in its area of market. All of the objectives and to it related strategies aim for a common goal. By reading through whole strategy process model we can assume multiple results.

One of them is, that even though there are few companies that are much bigger and well known in its close area, the company we work for is slowly gaining its reputation and loyal customers. Not only that, but company is also well known for creating products of very high quality for affordable price. For that, there is no doubt, that one of goals for the company will be to spread its awareness over its area through social media, since this method, if correctly used, can massively increase customer base and even create new possible opportunities for creating new contracts. This way, the company will increase interest of customers, which will passively increase overall avenue.

The second important goal for whole strategy is expanding company. Our group quickly assumed that this step will require a lot of monitoring as well as lot of patience and strategizing. Many companies in attempts of trying to expand ended up failing miserably. There are several examples of such companies, a great example belongs to, for example, XciteLogic [2], which through one year increased its size by 600% as well as hiring over 80 new employees. Their overconfidence continued with very fast and aggressive growth, with a lot of very bad decisions, which ironically caused their downfall and by the end of the year, the whole company bankrupted. These example shows vulnerability of this step. Keeping track of changing market, awareness of your clients, connections as well as rivals is key to successful growth.

Vision and mission

Defining proper vision and mission is an important step for each company. It is not only to set a clear purpose and goal of the company, but also to remind in the future what the company stands for. It can also help employees to understand and trust the company they are working for. It also helps developers to understand the company, which increases chances of coming up with a better and more suitable system.

Vision:
Delivering high-quality products quickly and efficiently.
Mission:
Due to high income for a small company the mission is to safely expand.

Stakeholder analysis

Stakeholder analysis summarizes all people directly or indirectly involved in the project, their contribution, interest in the project, influence on it and their importance. It is important to know all the people influenced by the development of the new system and their attitude towards it. With this knowledge, their reaction and expectations can be better estimated. From this, more appropriate schedule and planning can be done, which will help the project to succeed.

Stakeholders	Contribution	Interest	Influence/attitude	Importance
Owner	Money/Information/Decisions/Feedback	Increase the effectivity of the company	High/Positive	High
Misfits	New IT-system	Provide a high-quality system	High/Positive	High
Customer	Feedback	Products/Better service	Medium/Positive	Medium
Accountant	Feedback	Better working environment	Medium/Positive	Medium
Suppliers	Supplies	Improving sales and reputation	Low/Positive	Low

Fig.3 , Stakeholder Analysis

Since MS Stahl Milling consists of only two employees, the amount of stakeholders isn't very big. Besides us as developers and the owner, who requested a new system, interest in the new system can have customers, suppliers and the second employee of the company. Since it will help everyone and make their job faster as well as easier, there is no one with a negative attitude towards this project and all the stakeholders will benefit from it.

Preliminary study conclusion

The company is successful and created a name within this industry, but as our preliminary study shows, there are many threats that can endanger MS Stahl Milling. Many other smaller or bigger companies can substitute it. In order to grow and stay competitive, MS Stahl Milling needs to invest in a system which will help the company to take care of all of the data and fasten up the whole production process. As stakeholder analysis showed, right now there is no one who could negatively affect development of the new system and it is always more effective and time saving teaching new employees on the new system, since it will be vital for the company. While hiring new employees, the company should be careful with setting structure and creating desired culture, because hiring too many of them has ruined a lot of companies. MS Stahl Milling has currently a promising future and even with the threats there are many opportunities the company has.

Business modelling

Business case

Business case defines the project. It clearly sets what will be done within the project and why will it be done. Normally it states also how long and how much money the project will take, but considering this is a semester project, time for the project was pre-set and we cannot talk about financial costs, as this is part of our study. Since the business case sums up all the necessary information of the project, for the client it is a proposal from the developers. On the other hand, developers use it as a guidance to know what needs to be done within the project.

MS Stahl Milling is a company that specialises in producing precision milled components mostly for other companies like car, doors, electronics manufacturers etc. Currently they are using only pen

and paper, which slows down the whole working process and makes creation of statistics nearly impossible. Our team would like to implement a system for this company as part of our study case.

Strategic alignments

As a matter of fact, strategic alignment is one of the main key differences between organizations that perform well, while others don't. For such reasons, we can already assume that strategic alignment is critical to any organization that wants to achieve its objectives/goals and to outperform their rivals. Many companies underestimate the importance of the strategic alignments [3]. This argument can be backed by numbers, which say, that 10% of all companies manage to execute their plans correctly and to achieve their goal. Reason for that is that the rest of 90% didn't use any kind of strategic alignment. Strategic alignment table shows direct links between their strategy goals and the action they are about to take. Strategy can always change depending on specialization and mission of certain companies. For our company, we can assume that the strategic alignment table is pretty simple with very basic goals. Important task before starting to write alignment is to ask basic questions for the mission of the company. In our case, the question was clear: “How can the company manage to expand safely”? It is exactly for that reason why the final version of strategic alignments looks as it looks. At first view it may seem very basic, but it has its justification. Through these smaller steps we can ensure that the company will manage to safely control the balance between expanding and having financial backup.

Plan	Goals/Objectives	Relationship to project
2020 Managers strategic plan for inventory management system	Create new system for the company	Since a company doesn't possess any kind of system, it quickly becomes a very higher priority. New system will rapidly increase efficiency as well as finances for a slowly growing company.
2020 Managers strategic plan for inventory management system	Spread overall awareness regarding company within specific area	If a company wants to contest against other bigger companies within the market, this approach will prove quite effective (social media, new relationship,...)
2020 Managers strategic plan for inventory management system	Expand the main department of company	For company to grow, they should suggest consider accepting new employees as well as buying more production machines

Fig.4 , Strategic alignements

Product features

Since a business case is a proposal for a company, clearly setting all the features of the product is vital. This way, the customers can easily see what they should expect from the new system, which

the developers will build. For the developers, it sums up all the things they need to implement and think about.

Feature	Description	Priority
K1	The system does create, read, update and delete information of a order	High
K2	The system does create, read, update and delete information of a offer	High
K3	The system does create, read, update and delete information of a product	High
K4	The system does create, read, update and delete information of a customer	Medium
K5	The system does create, read, update and delete information of an employee	Medium
K6	The system does create, read, update and delete information of a procedure	Medium
K7	The system does create, read, update and delete information of an equipment	Medium
K8	The system does create, read, update and delete information of a supplier	Medium
K9	The system does use clearance level, allowing different access to different users	Medium
K10	The system does generate invoices	Medium
K11	The system does create statistics	Medium
K12	The system does schedule work	Medium
K13	The system shall be user friendly	High
K14	The system shall be error preventive	High
K15	The system shall be fast and responsive	High
K16	The system shall be easy to remember	High
K17	The system shall have a database to store all the data	High

Fig.5, Product features

Risk analysis

Risk analysis is used to analyse what can slow down the project and find a solution as fast as possible. The biggest threat for this project was naturally Covid-19 pandemic. Since it strongly limited our possibilities, all we could do is to take it into consideration and to try our best to overcome it. We are new to projects of this scale, so probability that we will encounter something unknown is very high. That's why we recognised our inexperience as second highest risk and planned to work a bit ahead just to ensure the success of this project. Already at the beginning of this project some of us were a little bit sick, so taking it into consideration is very important. It happens very often that interviews with the customers are misinterpreted or understood differently, therefore communicating with customers more often and coming up with proper questions for the interview is an important part of each project. Lastly, since a used system can fail from time to time, we recognise it as a risk we need to think about. This last problem has an easy solution, since always keeping backup will solve most of the issues caused.

Risk	Probability	Impact	Priority	Solution
Covid-19	5	4	20	Try hard to overcome this situation and communication barrier
Resources are inexperienced	4	4	16	Extra buffer in case problems arrive
IT staff sickness	4	3	12	Take sickness into consideration while creating plan and try to overcome it
Requirement is not precise	3	4	12	Discuss requirements and contact company if necessary
Failure of used system	3	3	9	Keep copy of each part of the project

Fig.6 , Risk Analysis

Domain model

Domain model is a visual representation of real-life “entities” in a model and relations between them. Each “entity” is represented as a class with attributes, which each object of a certain class possesses. It also shows which classes interact with each other. The number on the line called multiplicity shows how many instances of a certain class can be associated with how many instances from another class.

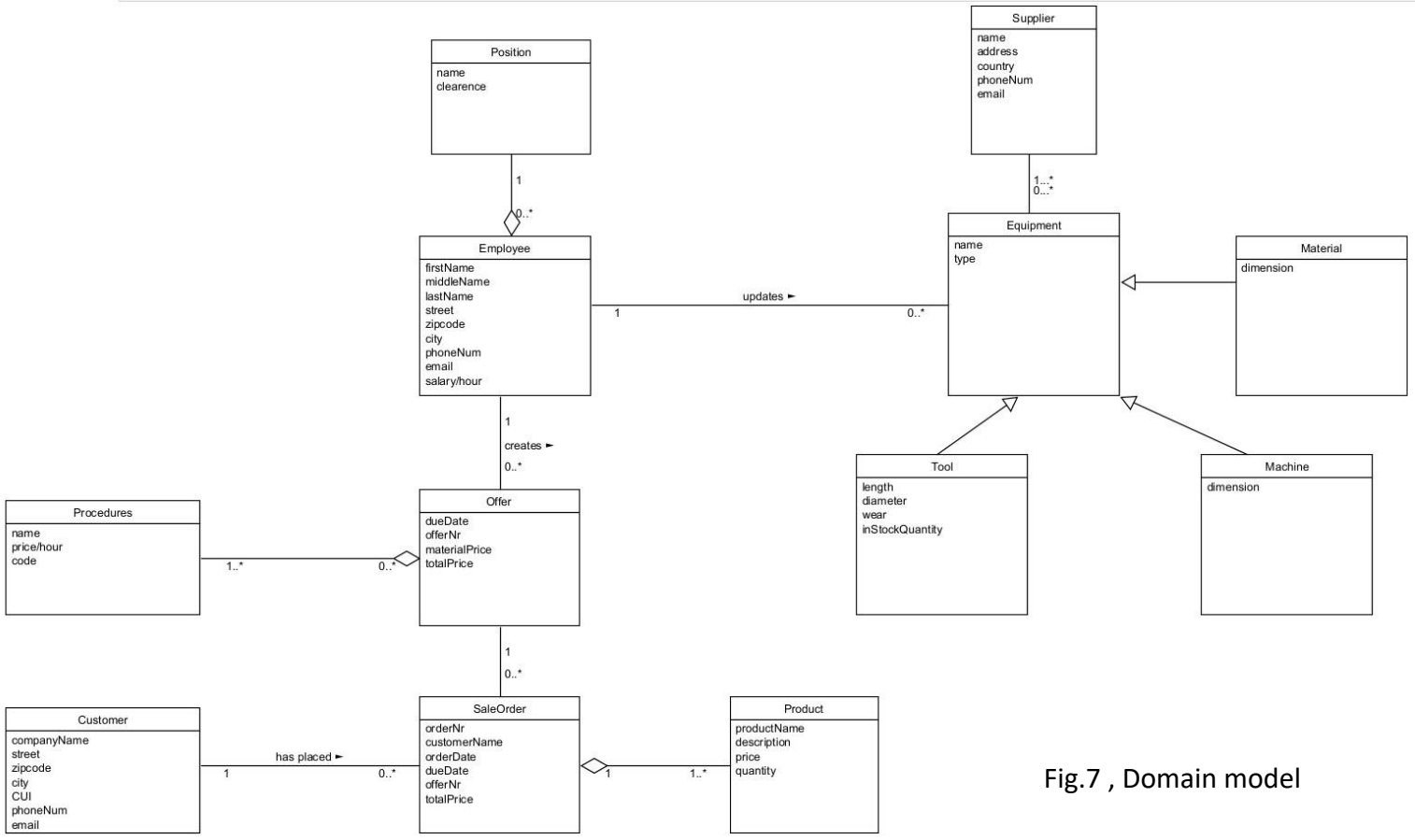


Fig.7 , Domain model

Figure 7 represents the domain model of our system. Within there each customer can place multiple orders. Since MS Stahl Milling is a B2B company, all the customers will be companies, that is why it has a field companyName. CUI is a Tax Identification Number in Romanian. Orders created by the company are called SaleOrders in our system to ensure compatibility with the database, since we could not create a table or field called order. Orders consist of one to multiple products and are associated with exactly one offer. Since each product is unique its instance can be associated with only one order and it has directly an attribute quantity.

Since the company can create multiple offers but some of them don't have to be accepted by their customers, each offer can be associated to one or zero saleorders. In order to know who created the offer, offers are associated with one employee and consist of one or many procedures. In most of the cases, from one offer only one order is created, but since it happened already that a customer was asking for the same order multiple times, there can be multiple orders created for one offer in our system.

Each employee is assigned a position with a certain clearance level, which will allow or decline access to certain parts of the system. Depending on that, employees can create offers and update equipment. Every equipment has at least one supplier, but there can be multiple suppliers assigned to one equipment.

Requirements

System vision

Currently the MS Stahl Milling is working only with pen and paper. It is essential for a company to have a reliable system with which they will be working in order to grow and stay competitive. The existing manual writing on papers is slow and can lead to many complications such as losing essential information or making it nearly impossible to create overall statistics.

Purpose of this system is to make the job of future and current employees easier and more effective, easily dividing work while also saving all the important data. The system must support the order process as is displayed in Fig.7 and later in Fig. 11, schedule all the work between the employees and store every data within the database.

Right now the users will be only the owner and the accountant. Since only a few parts of the system are relevant for the accountant, the system must be developed in close cooperation especially with the owner. Since the IT skills of future employees will vary and cannot be predicted, the development of the system needs to rely more on the owner, his experiences and rather assume lower IT skills of the users.

The whole system will be created in Java using Eclipse as IDE, for the graphical user interface Java Swing will be used and MS-SQL to store all the data in the database.

Mockups

Mockups are like the first idea of how our system will look like at the end of the development. Even though in the finished version it is mostly different, mock-ups are created to find important aspects which have to be considered, agree on some conventions while creating GUI and during

think-aloud testing to discuss with the customer if it is user friendly, if he can find everything he needs etc.

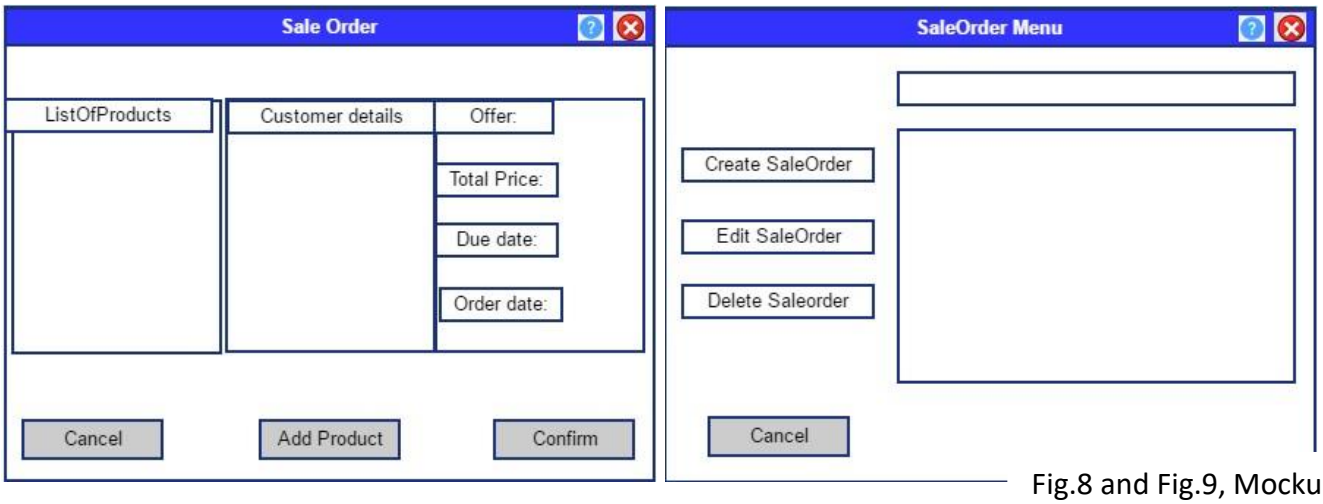


Fig.8 and Fig.9, Mockups

As for our mock ups, we created few ones, but our most important ones are for creating orders. Fig.9 shows menu, on the right user can see all the orders and search through them using the window above. Clicking on any specific order, user can then click on edit or delete button, which will either delete it or allow the user to edit it. Clicking on the cancel will get the user back to the main menu. The question mark will pop up a help window which will describe current window.

After clicking the “Create SaleOrder” button, “Sale Order” menu will pop up. This menu consist of three sections: left, which shows all the products added to the order, middle one shows information about the customer and the left one shows information about the offer, from which the order is created. To add products to the order, the user will use “Add Product” button, which will open up a window, where new product will be created. The user can either abandon the creation with “Cancel” button or finish it with “Confirm” button.

Once we started working on the graphical user interface(GUI), we realised that we need will not be able to make it completely according our mockups. First of all we changed the colours for a bit more fitting once for this company. The Menu from Fig.9 nearly as in our mockups, only with small changes in button naming. We have used this design also for other similar menus to keep it more consistent and familiar. Most of the changes where on Fig.8. While creating new order, we realised all customer details are unnecessary and keeping only customer name is more than enough. Secondly this look is chaotic, so something clear and more simple will better. Omitting the middle part of Fig.8 and organizing it more appropriately while still using all the other parts is the way we created our current “Create Order” menu.

Use case model

As it is written already in the Unified Process section, this process is use case driven. Use case model describes proposed functionality of overall system. As such, it mostly describes the goals of users (actors), who will interact within the system, the interactions between the users and the system, as well as, for example, required behaviour of our system to satisfy the needs of users. As for the actor, they may represent roles played by humans, external hardware or even other

subjects. Actors are always outside of the system, interacting with it by providing input, or simply receiving outputs from the system.

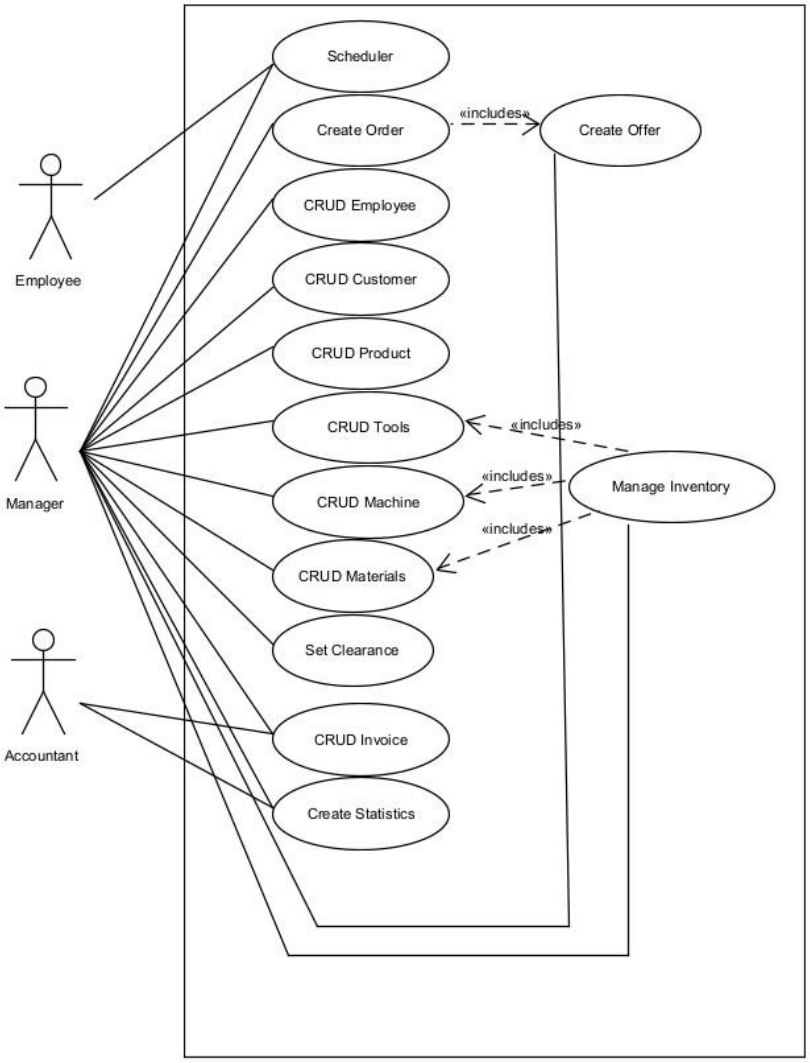


Fig.10 , Use case model

Use Case Diagram of our system is shown in Figure 10. Within our system, there are 3 actors: employee, accountant and manager. As the diagram shows, the employee will be only using scheduler, since the scheduler will show all the work that needs to be done in a certain day. Next actor of our use case model, namely the accountant, will be mostly only generating invoices and statistics.

The managers will be using our whole system. They will be creating new orders, which first start as an offer, that is why we included use case “Create Offer” within “Create Order”. Equipment of the company, namely tools, machines and materials are part of their inventory, and since the work with them (meaning ordering new, updating current, adding new etc.) will be very similar, we decided to include them in one use case called “Manage Inventory”. Use cases are described in brief use case description and more elaborate in fully dressed use cases, which is mode for the most complex uses cases.

Use case: create Offer
On customers behalf, the employee creates an offer for the customers order and it estimated price.

Use case: create Order
A customer asks for a new order. An employee creates new order with all the products needed, refering to the already accepted offer.

Use case: create Statistics
User uses the system to generate statistics regarding customers, employees, offers and orders.

Use case: set Clearance
User will set clearance level for employees, allowing and disallowing them to use certain parts of the system.

Fig.11 , Fully dressed use

Use case name: create Order	
Actors: Manager	
Pre-conditions: Company requests a new order and offer was accepted, the offer and the customer exist in the database	
Post-conditions: Order succesfully created and stored in the database	
Main Success Scenario:	
Actor(Action)	System(Response)
1. Manager wants to create new order.	2. System returns all the offers from the database.
3. Manager chooses an offer based on which the order will be created.	4. System returns all customers and product names.
5. Manager chooses an customer.	
6. Manager adds a product to the order by writing all the product details.	7.System adds product to the list.
8.Steps 6 and 7 are repeated until all the products are added.	
9. Manager confirms the order.	10.System creates the order with all products and stores all the products and the order in the database.
Alternative flows: 3a: There are no offers in the database. 1. User will create the offer and then creates new order again. 5a: There are no customers in the database/desired customer not found 1. User creates the the customer in the system and starts creating order again from step 1.	

As it will be soon mentioned, create Order is the use case with highest importance, both by business value as well as within architectural importance. Not only that, but creating Order will belong to one of our most complex use cases in the whole system. For exactly these reasons it is necessary to create and describe a fully-dressed use case. Fully-dressed use case unlike normal use case includes way more details and are structured as well. What is more, thanks to fully dressed use cases, we will manage to gain better understanding of the goals, tasks and requirements.

	Risk	Coverage	Criticality
Create Order	Medium	High	High
Scheduler	High	Low	Medium
Manage Inventory	Medium	Low	Medium
Create Statistics	Low	Low	Medium
Set Clearance	Medium	Low	Low

	Business value	Architectural importance	Total
Create Order	10	8	80
Scheduler	5	6	30
Manage Inventory	7	4	28
Create Statistics	5	3	15
Set Clearance	3	4	12

Fig.12 , Use case priority table

According to unified process, developers will start first with the highest priority use case and then, they will continue with lower and lower priority. Point of this is to omit the risk of being unable to finish the project. Once it is possible to implement the most complex and important use cases, the developers will be surely able to implement the whole system. If it is not possible, at least the project will fail at an early stage, resulting in much lower loss. Priority of use cases is evaluated based on risk, coverage and criticality or business value and architectural importance. Risk means how hard it is to implement the use case, coverage how big part of the system it is/how many parts of the whole system it touches and criticality how important is the use case for the company. Business value is basically return of investment, how important is this part of the system for the company and architectural value is how important is the use case for the development of the whole system.

Since use case “Create Order” has the biggest return of investment, meaning it has the biggest business value/criticality, it touches nearly all the parts of our system so it has highest coverage/architectural value and it is complex, since it consists of two parts, we decided to make it our first priority use case. On the second priority we have put the “Scheduler”. Even though it is not necessary for running the business, it is the most complex use case consisting of parts we had no experience with. After that managing inventory is important for every company, so we recognised it as a third priority, then creating statistics since it has big business value/criticality and

lastly setting clearance. Even though setting clearance can be a bit complex, it is not that important for the company.

Analysis and Design

Relational model

The relational model is created to have an overview of how the database, specifically tables in the database, will look like. In the relational model, you can see foreign keys, which are attributes referencing other attributes in different tables and primary keys (underlined attributes), which uniquely identify that specific table. Basically said each class in the domain model (Fig.8) will have its representative in the database. Since the size of this model makes it unreadable in the report, in this part we will include and describe only a few parts and include the rest within the appendix 1 and hand in material.

Fig.13

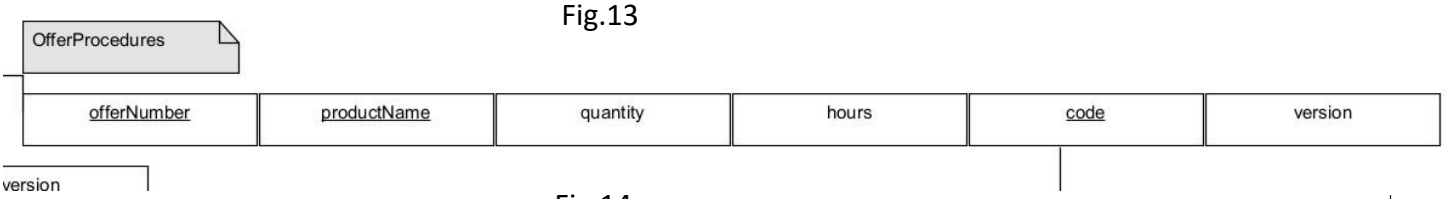
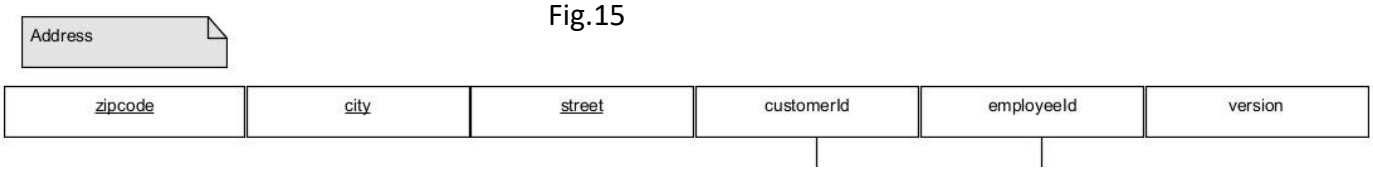


Fig.14

Fig.15



Each part consists of all the attributes from the domain model with an extra id attribute which will be auto generated in the database and version which is described in the section “Database Concurrency”. Fig.13 is a special table for connecting to tables between which there is a many to many relationship within the domain model. Normally it only consists of primary keys from both tables, which is different in our case. We needed to add extra fields because of the scheduler, which is not part of the domain model but needs this data to be connected. Even though it looks like it connects offers, procedures and product tables, it is not the case. In this case, we could not refer “productName” to the product in the database, because in this part the user only creates an offer for their customer, so product at that point is not a thing, it is just an abstract name and quantity. But it was necessary for the functionality of the scheduler.
Since the address has multiple values, we decided to separate it and create a table address with customerId or employeeId as foreign key. Since we use it for customer and employee as well, there will be some null values, which isn't the best practise, but in this case it is better to have few nulls than to have exactly the same table with only one different value.

After creating our relational model, we needed to formally validate our tables using the normalization, namely first, second and third normal form. First normal form says that all the

attributes must be atomic, meaning that none can be multivalued. In our tables, the only thing that could be considered multivalued is having multiple phone numbers or emails in Fig.Y, but we do not consider them nor allow them to be multi valued.
Second normal form says that all non-key attributes must be dependent on the whole key. In our relational model, this rule is not applied only in scheduler necessary parts of the database, as can be seen in Fig.13. Only in this case we had to omit it, since it was necessary. Otherwise, we would have to create a much larger amount of tables, making it nearly impossible to work with this database. The data would be split among too many tables, which makes it more reasonable to not use this rule this time.
After having the relational model in second normalization form, a third normalization form can be applied. Third normalization form says that all non-key attributes must depend on the primary key only. This is applicable in our whole relational model, except scheduler related tables which are not even in second normalization form. Arguable could be city and zipcode relation in fig.15, since we could say that zip code depends on the city and street or that city depends on the zip code. This is also the place where we decided to omit the normalisation rule because creating separate table only for one or two values would complicate work with it a lot without it being useful in any way.

Architecture

Architecture defines systems structure. Using a proper architecture helps the system to be more maintainable and understandable. In this project we were using the layered architecture. Our layer architecture consists of four parts: presentation layer(user interface), controller layer, model layer and database layer.
Presentation layer handles interaction between the user and the system.
Controller layer contains the business logic. Usually there is one controller class for each use case.
Model layer consists of domain entities. It is created from the domain model.
Database layer consists of classes interacting with the database.

Layered architecture has multiple pros and cons. Since the logic and placement of classes is clear, this architecture is readable and easy to follow. Because of that, we are also separating the concerns, reducing coupling and dependencies and increasing cohesion. Another benefit of this architecture is the ability to change the presentation layer with only a little amount of coding necessary, or reusability of certain layers.
On the other hand adding new features to the system means all the layers will be affected, which can be complicated in bigger systems. When reaching classes in lower levels, the system needs to go through the whole system, which can decrease the performance.

SSDs and Contracts

System sequence diagrams show how the user interacts with the system and how the system responds to their actions. Those are created from fully dressed use cases.

Contracts are created for parts of the SSD, where change in the system is happening. In our case, this happens only when the user confirms creating the order. After that, all the data, meaning products and order are stored in the database.

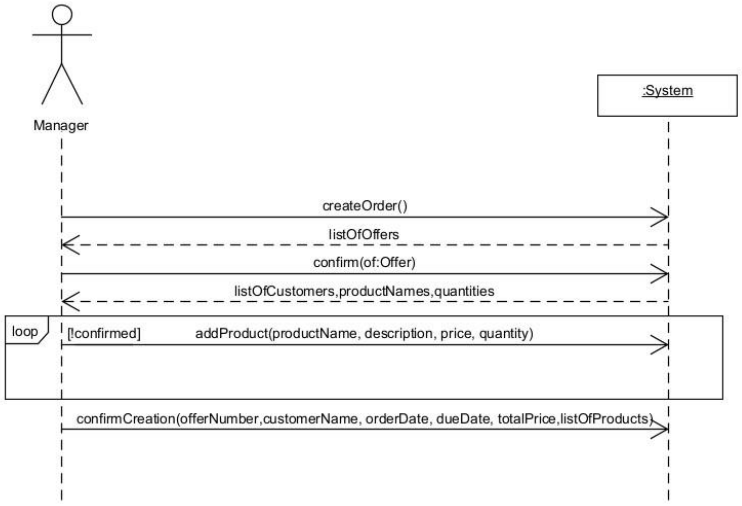


Fig.16, SSD

Contract:
Cross Reference: create Order Operation: confirmCreation(offerNumber,customerName, orderDate, dueDate, totalPrice,listOfProducts) Precondition: Corresponding offer exists in the database and was chosen. Customer exists in the database and was chosen. Postcondition: Order is created. Offer is assigned to the order. Products are assigned to the order. Order and products are stored within the database.

Fig.17, contract

Interaction diagram

Interaction diagram serves as the model that describes how objects collaborate between each other. It shows how instances of classes are created and how these are interacting between each other. There are two types of interaction diagrams: communication diagrams and sequence diagrams. For this project we decided to use a sequence diagram. Sequence diagram may be larger than the communication one, but is much more readable and easier to understand. We have used sequence diagrams already many times and prefer to work with it much more.

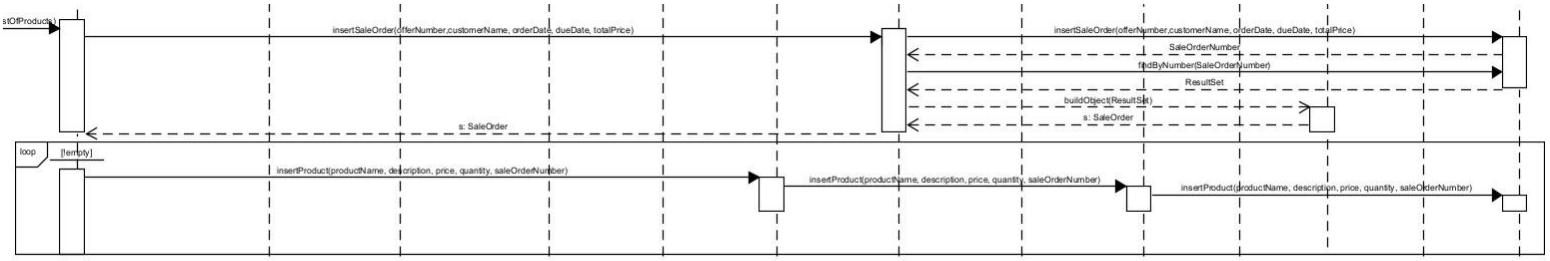


Fig.18, Interaction diagram

The sequence diagram is really big in size, so it would not fit directly in the report. That is why we included it as an appendix 2 as well as part of our hand alongside with code and script and in this section describe only one part of it. In this part, the user confirms creation of the order and the order with products is saved in the database. We first save order in the database and directly retrieve its “SaleOrderNumber”, which is autogenerated. Then we search for order with this number and build an object SaleOrder, which we pass back to the “SaleOrderController”. Thanks to this we can check if the order was successfully stored in the database. For the database, we also

need autogenerated “SaleOrderNumber” for all the newly created products, which will be saved alongside with this number. So after getting the the SaleOrder object in the “SaleOrderController”, program stores all the newly created products one by one, hence the loop, alongside with “SaleOrderNumber” in the database using the “ProductController” and “ProductDb” classes.

This actually is not our first official version of the interaction diagram for the use case “Create Order”. We had to make changes and refine this diagram after implementing a graphical user interface, since there were some features we used in order to have it more clear and user friendly. Because of that, creation of object order and retrieving and saving data had to be changed, which resulted in changing the diagrams.

Scheduler

Since the scheduler was a new and complex part of the system that we didn’t deal with before we took in consideration the fact that diagrams would change a lot and the pandemic situation that will slow the work drastically so we decided to take a classic approach in designing it thus we used the old pen and paper to create diagrams that will help with the implementation.

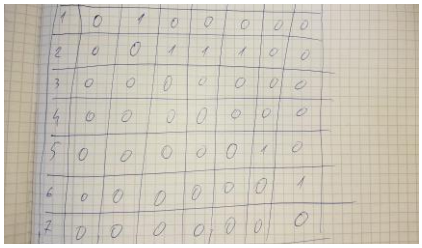


Fig.19

Fig.19 represents a matrix used to store priorities between procedures.It corresponds to an oriented graph.

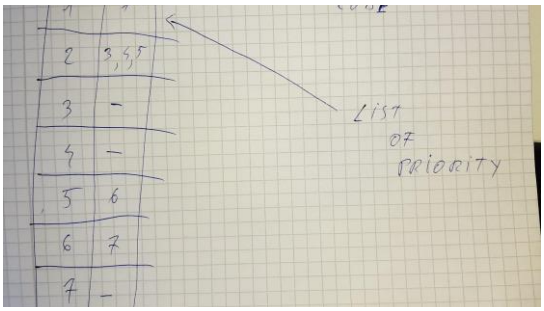


Fig.20

Fig.20 represents a temporary list of one to many that will help order the procedures.

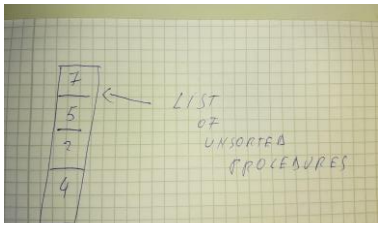


Fig.21

Fig.21 represents the list of unordered procedures received from schedulerCOntroller.

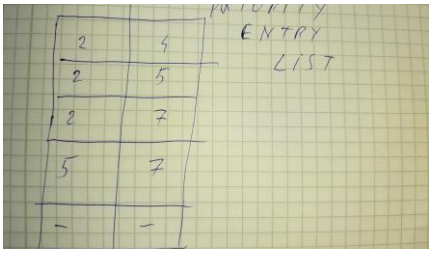


Fig.22

Fig.22 represents a list of one to one priority where the second column procedure requires the previous pair to be done before it can be done itself.

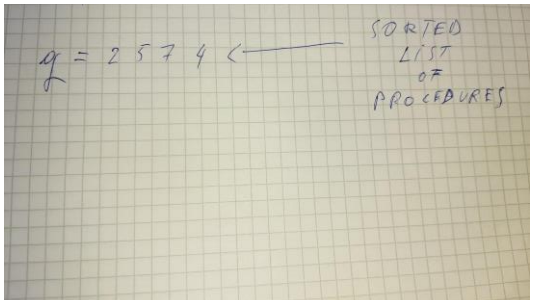


Fig.23

Fig.23 represents the ordered list that is returned after sorting.

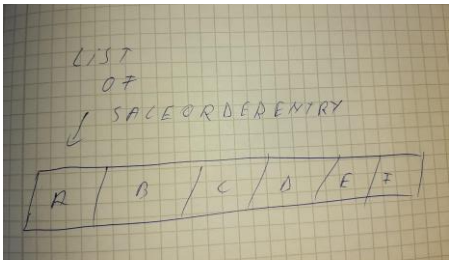


Fig.24

Fig.24 represents an unsorted list of Orders and the products with their respective procedures stored as saleOrderEntry.

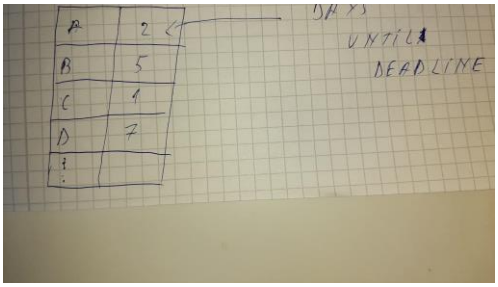


Fig.25

Fig.25 represents a map of orders and the amount of days until their deadline.

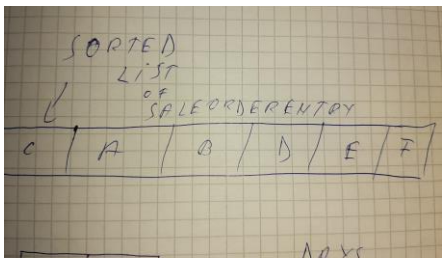


Fig.26

Fig.26 represents a sorted list of saleorderEntrys.

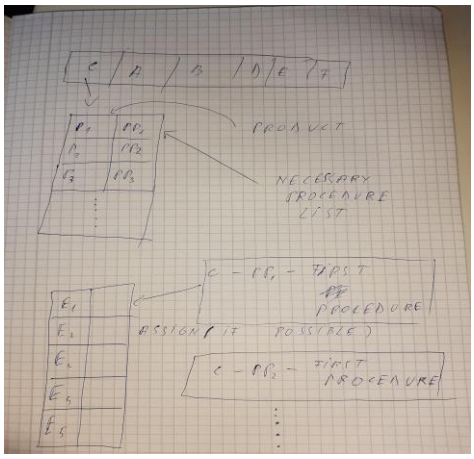


Fig.27

Fig.27 represents the process of assigning a procedure to an employee.

Design class diagram

Design class diagrams display all the classes, methods and fields in the system, return types and which classes access which. It is the most complex diagram and in bigger systems like ours, it is so big it will not even fit in the appendix, so we will include part of it in the appendix 3, the whole diagram in the hand in and fraction here.

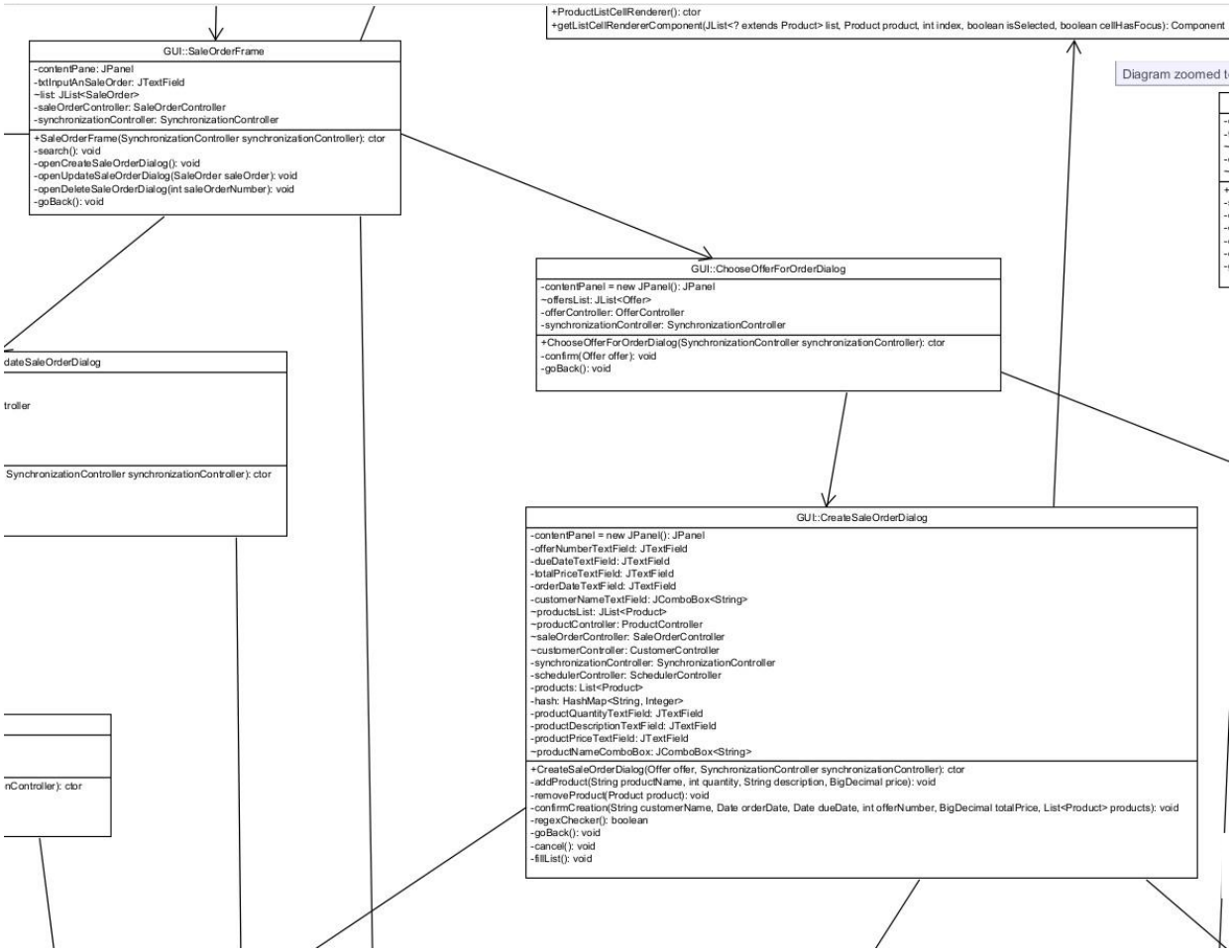


Fig.28

Figure. 28 shows graphical user interface part of order creation. First the “SaleOrderFrame” will create a “ChooseOfferForOrderDialog” class, where all the offers will be displayed to the user. After choosing an offer, “CreateSaleOrderDialog” class will be created, where the user will choose the customer and add all the products needed for the order. Both of these classes will call the classes from the controller layer, since all the logic is stored there.

Implementation

Code Standard

Implementation of the code required us to agree upon a code standard to make it 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[4]. By following them, we properly used spacing, indentation and other good practices. Method names should state their function in the system. In correlation with the newly learned material during second semester, we tried to use new data structures that we aren’t that experienced with. Wherever we could, we tried to use lambdas.

```
JButton goBackButton = new JButton("Go Back");
goBackButton.addActionListener(e -> goBack());
```

```
public HashMap<Employee, Procedure> getSchedule() {

                HashMap<Employee, Procedure> tempMap = new HashMap<>();
                scheduleMap.forEach((k, v) -> {
                        employeeList.forEach(e -> {
                                if (e.getId() == k) {
                                        try {
                                                tempMap.put(e, procedureController.findProcedureByCode(v));
                                        } catch (DataAccessException e1) {
                                                System.out.println(e1.getMessage());
                                                e1.printStackTrace();
                                        }
                                }
                        });
                });
                return tempMap;
        }
```

DAO Pattern

For the interaction with the database we are using DAO pattern. Data access object pattern (DAO) is a pattern that allows abstraction of persistence in an application. It is an intermediate class that helps the system to maintain low coupling and also it prevents the system from directly accessing and modifying data in the database ensuring database integrity.

Scheduler

The scheduler consists of two main parts, first one responsible for ordering procedures which was created under the ProcedureOD implementation and the other that orders the saleOrders and assigns procedures to employees which was created under the Scheduler implementation also being the main body of the use case implementation.

For the most part employees can't interact with the scheduler the exception being notifying it that the assigned task has been fulfilled.

ProcedureOD is the part that orders the procedures in a specified list.

The procedureODController starts by retrieving the matrix that represents the oriented graph containing procedure priorities see Fig.19. After that it goes through the matrix and creates a map containing the code of each procedure and a list of procedures that need to be done after that specific one see Fig.20. It uses the procedureList that was given by the schedulerController see Fig.21and recursively searches the list Fig.20 creating a separate entry for each connection and stores it in a list see Fig.22. After that it takes each procedure and compares it to each entry if no need for a previous procedure to be done before was found or that procedure is already listed for a schedule it will list that entry as well in the list that will be returned see Fig.23. After that it's job is done.

The schedulerController starts by retrieving the previous assignments for employees and all not done or in work orders with priorities see Fig.24. Then it creates a list of orders and their remaining days until deadline see Fig.25 which it uses to sort the order putting the most urgent one to be scheduled first. After that it iterates thru all orders form the most urgent one to the least as long as there are employees with unassigned tasks, it takes the first procedure from each product in one order and looks if the employee can do the specified task if the employee is able to perform it than

it assigns it and marks the procedure as being worked on before moving to the next free employee see Fig.27.

If an employee has finished it's task it will go to the scheduler interface select his name and click done at which point it will mark the procedure form being worked on to done and call the scheduler to assign a new task to him and any potential employee that could have been waiting for that task to be done so he can start on the next one.

The scheduler is called mostly by the system only when it is needed.

Scheduling is an intensive task that can take quite a bit of resources as such it doesn't run all the time and when it is needed it runs on a separate thread in the background so as not to inconvenience the user.

Parallelism and Concurrency

When we talk about concurrency, it can either be seen in the database, in the back-end or even in the front-end. Concurrency or namely multi-threading is primarily used to lessen the load on the single core of a processor that the code in most languages usually runs on only. This allows the work to be spread between the several cores of the processor which greatly improves processing times and may result in better CPU temperatures. This can be achieved in Java by using threads, creating them, putting the more expensive and time-consuming code for them to run and finally starting them. In the field of databases concurrency already exists, but the developer has to make sure several users can read and modify the same data at the same time without any issues. And finally in the front-end it can be seen used for rendering some heavier and more memory-consuming components in the background.
In the world of concurrency we also have parallelism. It is the division of a problem, which has to be done, into several smaller subproblems and all of them ran in separate threads at the same time - in parallel. At the end the results of the solutions to the subproblems are combined into one. An example of parallelism is the stream which can be run either in serial or in parallel. If executed in parallel the Java runtime partitions the stream into multiple substreams. Aggregate operations iterate over and process these substreams in parallel and then combine the results. So in other words it provides a great boost in overall performance, but this is mostly true when we are dealing with big amounts of data.

During our project we encountered several concurrency related problems which we had to tackle and beat.

One of them was running a scheduler that takes much more time to finish in comparison to other tasks. We solved this problem by runni

Problem № 1 - Database Concurrency

Solution: Optimistic approach

Database

For the work with the database are often used so-called transactions. The transaction is a set of statements that has to be executed as a unit. When something goes wrong and any part of the transaction cannot be executed, then this transaction is not executed at all. A typical example used to clarify transactions is money transfer between two accounts. The user wants to withdraw money from one account and deposit them on the other. This has to happen as a unit, it cannot happen that error in between will result in withdrawing money from one account but not depositing in the other one.

Since the database is accessed by many users, many conflicts can arise. This is not a problem when everyone only reads the data. but all other actions can result in different conflicts. Changing data can result in other users reading “old” data, which may not even exist anymore. These problems mostly occur in transactions, since transaction consists of multiple statements.

Problem handling

Basically there are two approaches when dealing with concurrency in the database: pessimistic approach and optimistic approach. In the pessimistic approach, we assume that conflicts will arise. We expect there will be many conflicts. In this case, we lock the data we are working with so no other user can access that particular data at that time. To increase the effectiveness there are different locks depending on what the user is doing with the data. For example, if there are multiple users, who are only reading the data, there is no reason to restrict them the access since the data will not change. This locking is secured by different isolation levels. Although they prevent conflicts to happen, their drawback is lower system responsiveness. The stronger the isolation level, the lower is the responsiveness. They need to be chosen carefully in order to stabilize the system. Using locks creates another problem, namely a chance of getting into deadlock. But in the system, where many conflicts can arise in the database concurrency, this is the best way to prevent them.

The optimistic approach is the polar opposite. We assume that the probability of concurrency problems is very low and they will presumably never occur. It does not mean that the programmer won’t do anything and blindly believes that problems will never happen. This only means that no locks will be used. Even though very small, the possibility of conflicts still exists. That is why in the optimistic approach there is a thing called validation. The validation ensures no conflicts will arise. The biggest advantage of this approach is the system’s responsiveness. On the other hand, if there would be any more conflicts, mostly it would mean more repetitive work for the user, since, during the conflict of two users, one would have to repeat his actions all over again.

Long discussions of our group came to a conclusion, that the best for our project will be to take the optimistic approach. A fast and responsive system is naturally required, but the reason for our conclusion is the unique way of the company's operation. Every time company has a new order, they create a unique object specific for that order. The probability of having two orders with the same product is nearly non-existent. Restocking does not happen very often and again since each product needs different materials and handling, the probability of getting into conflict with multiple users is extremely low, even if the company would grow and expand. One way to come into a problem is for example when two people would brake exactly the same drill at the same. The database in our system is mostly used for storing data, reading and creating statistics, adding new data but rarely updating old ones. That is the most important reason, why we decided to take the optimistic approach.

Implementation

Unlike the pessimistic approach, the optimistic one requires an extra bit of code to be written by the developer, both in the database and in the program. Fortunately for us, the solution wasn't complex at all.

```
create table customer
customerId int identity(1,1),
companyName varchar(30) not null,
CUI varchar(30) not null,
phoneNumber varchar(30) not null,
email varchar(30) not null,
version BIGINT not null
primary key (customerId));
```

We will take for example one of the tables we have for our system - the Customer shows the script used for inserting the Customer table into the database. Only a single line of code is needed to be added here - the Version field, which is of type Int. Whenever a new Customer is created the Version field is set to 0.

```
update dbo.Customer set companyName = ?, CUI = ?, phoneNumber = ?, email = ?, version = ? where customerId = ? and version = ?;
```

```
for (int i = 0; i <= 5; i++) {
        try {
                dbConnection.startTransaction();
                int version = findVersion(customer.getId());

                updateCustomer.setString(1, customer.getCompanyName());
                updateCustomer.setString(2, customer.getCUI());
                updateCustomer.setString(3, customer.getPhoneNumber());
                updateCustomer.setString(4, customer.getEmail());
                updateCustomer.setInt(5, version + 1);
                updateCustomer.setInt(6, customer.getId());
                updateCustomer.setInt(7, version);

                res += updateCustomer.executeUpdate();
```

The best example where the Version field is of use is an update method of any given class - in our case the Customer. Update means modifying data so if multiple users do it at the same time, it can become problematic. So in order to ensure safety we use the Version field together with a transaction. After the transaction starts, the method retrieves the current version of a existing Customer in the database, after that all the updated field from the GUI are passed to the PreparedStatement and finally the PreparedStatement is executed. At the end is where the check is done, if the Version is different than when the transaction began, the transaction rollbacks and tries again. After the transaction successfully commits the changes, the Version field of the Customer is incremented by one.

Problem № 2 - GUI Concurrency(Dynamic refresh and Heavy tasks)

Solution: Synchronized Threads for Dynamic refresh
SwingWorker for Heavy tasks

Synopsis

Having a dynamically refreshing GUI, which doesn't wait for the user to manually do it everytime or have a bunch of timers going off and on, constantly making unnecessary checks, is a pretty big deal in most applications. Usually this is one of the most problematic parts of a program in terms of code complexity, since it's done entirely by creating and starting threads. But all this hardship can be partially avoided by using a monitor, which will take care of everything the threads want to do.

Implementation

The easiest solution to our problem we encountered is to use a monitor, which will basically 'monitor', control which thread needs to be let through and which thread has to be stopped to wait for the other ones to finish their task first. The monitor could be seen as a containment area for threads or a wall which decides who goes in and who goes out - in our case these are the threads. Implementation wise, we need the monitor which will be a new class and then in the rest of the program, the places where there is a list that needs to be updated, an additional internal private class has to be created, which will be spawning a thread, whose task is to update the list. As for the places where the user's input calls some methods that change in some way the content of a list somewhere else in the system, there isn't a need of a separate class, a simple call to the monitor object is enough to wake all other threads to do their job.
It is a key factor to use the same monitor object in all places. Since every object in java is considered a monitor of his own, all threads have to be synchronised to that specific object. If this isn't the case then we will run into many problems and errors, since each thread will be synchronised to a different monitor object, so they can't see each other. One solution is to just pass that one monitor object around to all classes that can make use of concurrency.

```
public class SynchronizationController {

        public synchronized void get() {
                try {
                        wait();
                } catch (InterruptedException e) {
                        e.printStackTrace();
                }
        }

        public synchronized void set() {
                notifyAll();
        }
```

In our case we have named the monitor object - SynchronizationController, since that is it's task. The whole class only consists of two synchronized public method that don't return anything - get and set. The role of 'get' is stop any incoming threads calling to this method by making them wait in the 'wait' method, until someone notifies them. And that's the role of the 'set' method, to notify any threads that are waiting on this specific object that they are free to leave and do their job.

```
class SaleOrderListThread extends Thread {
                @Override
                public void run() {
                        while (true) {
                                synchronizationController.get();
                                new GetSaleOrderListWorker().execute();
                        }
```

Now we will take as an example of creating a new class that extends thread - our SaleOrderListThread. It consists of a 'run' method which is necessary for the thread to be started, a infinite while loop in which the thread calls the aforementioned 'get' method of the monitor object and gets stuck there waiting to be notified and thirdly a new SwingWorker class is created and executed. The SwingWorker is a thread that is specific to the Swing API and what it does in short is runs some time-consuming and complex code in the background and returns a result, in our case it returns a list of sale orders.

```
saleOrderController.createSaleOrder(customerName, orderDate, dueDate, offerNumber, totalPrice, products);
schedulerController.schedule();
synchronizationController.set();
```

Finally we have GUI class that is for creating new sale orders. In the picture above it can be seen that after actually finishing the creation of the sale order, the 'set' method of the monitor object is called. This notifies all threads everywhere to leave the 'wait' method and continue.

Synopsis

A lot of the time the Swing GUI requires a lot of memory intensive and time-consuming tasks like getting data from database or iterating over several loops. Swing has three predefined threads - Initial Thread, Event Dispatch Thread(EDT) and Worker Thread. If the aforementioned tasks start, they will be executed in the EDT, which is really problematic, since the EDT works similarly like a

queue and that means that any other incoming task will be blocked by the slow ones, which will lead to the GUI freezing and being unresponsive. Creating and using an explicit thread is not an option, since Swing doesn't work well with them, thus resulting in unpredictable results. For that reason there is the third predefined thread, the Worker Thread. What is special about that thread is that it runs in the background and doesn't create a jam in the EDT. The only way to actually use that thread is by using a Swing Worker.

Implementation

```
class GetSaleOrderListWorker extends SwingWorker<DefaultListModel<SaleOrder>, Void> {
		int input = -1;

		public GetSaleOrderListWorker() {
		}

		public GetSaleOrderListWorker(int input) {
			this.input = input;
		}

		@Override
		protected DefaultListModel<SaleOrder> doInBackground() throws Exception {
			DefaultListModel<SaleOrder> listRepresentation = new DefaultListModel<>();
			List<SaleOrder> modelList = new ArrayList<SaleOrder>();
			if (input == -1) {
				modelList = saleOrderController.getAllSaleOrders();
			} else {
				modelList.add(saleOrderController.findSaleOrderByNumber(input));
			}
			for (SaleOrder so : modelList) {
				listRepresentation.addElement(so);
			}
			return listRepresentation;
		}

		protected void done() {
			try {
				list.setModel(get());
			} catch (Exception e) {
				e.printStackTrace();
			}
```

In order to create a new swing worker, one has to create an internal class that extends SwingWorker and set the two parameters. The first one is the return value after the worker is done with his task. The second one is the return type while the working is doing his task. The two methods that must be implemented are the 'doInBackground' and the 'done'. All the calls to database and loops are put in the first method, except GUI related methods. After that method is done, the second one is called. There it is safe to put GUI related methods like 'setModel' for example, since now it is executed in the EDT, and the result of the 'doInBackground' method can be retrieved by calling 'get' and then used.

Problem № 3 - Second Use Case - Scheduler(Heavy tasks)

Solution: Multi-threading for heavy and expensive tasks

One of the problems was running a scheduler that takes much more time to finish in comparison to other tasks. We solved this problem by running the scheduler in a separate thread that can do all it's intensive tasks at the same time as another part of the program without stopping the user or any other task until it has finished. Also in order to reduce the resources consumed by the system the scheduler doesn't constantly run unless it is needed.

Test

Integration testing

Integration testing tests interaction between objects. It tests the methods ,which are not dependent on single calls but multiple ones. They are derived from interaction diagrams, which show interactions between classes. In this way multiple methods are tested, because with the desired method are tested also its subsequences.

For the testing we were using JUnit 5, which differs in annotations from JUnit 4.
We have created integration tests for all our implemented use cases aside from the scheduler.

```
@Test
public void testUpdate() {
        try {
                returnValue.setTotalPrice(BigDecimal.valueOf(789.23));
                returnValue.setDueDate(Date.valueOf("2012-07-03"));
                returnValue.setCustomerName("Something");

                assertTrue(controller.updateSaleOrder(returnValue));
                returnValue = controller.findSaleOrderByNumber(1);
                assertEquals(BigDecimal.valueOf(789.23), returnValue.getTotalPrice());
                assertEquals(Date.valueOf("2012-07-03"), returnValue.getDueDate());
                assertEquals("Something", returnValue.getCustomerName());
        } catch (DataAccessException e) {
                e.printStackTrace();
                fail("Error: DataAccess exception");
        }
}
```

One of the tests we made is updating a SaleOrder in the database, which traverses from controller layer through database and model layer. Thanks to this test we can surely say that updating an order and interaction with the database works correctly, as well as our version control. Some tests we have, test multiple database classes at the same time. All other our tests are very similar, differentiating only on the parts tested and imputed test data.

Software Quality Assurance

Software quality assurance consists of multiple activities ensuring the quality of the development process. It is focused on the development process and preventing creation of errors. Basically it consists of planning on how the system will be developed, what precautions will be taken and how the errors and defects will be found and acting according to found defects and errors.

After implementing parts of the system, of which logic we clearly described in diagrams, multiple of us tested the implemented part to ensure its quality. WIth the implemented GUI we also tested especially usability and reliability. According to errors or possible improvements we found, we corrected the system to ensure the highest quality.

User Manual

After launching the application, a window with the title Main Menu appears on the screen. The program can be separated into 6 different sections, each of which can be accessed by the buttons in the Main Menu.

0) **Main Menu**

1) **Offer:**

1.1) Offer Menu:

- If the user wants to create a new offer, he clicks on the Create Offer Button.
- If the user wants to update or delete an existing offer, he needs to first select an offer from the list of offers, then the buttons will light up and be available for clicking.
- If the user wants to search for a specific offer, he writes a number, which corresponds to the offer number of an offer, in the field above the list and the specified offer will appear on the list.
- If the user wants to leave the current Offer Menu window, he clicks on the Go Back button and he will be returned back to the Main Menu.

1.2) Create Offer Menu:

- If the user wants to create new offer, he does the following:
    - Fill out the name of the product and how many of it the user wants.
    - The user will see a list of procedures. When he selects a procedure from it, he will have to fill out the Hours field on the right before he can assign the procedure to the product. After that the button Add Procedure will light up. Pressing it assigns the procedure to the product.
    - The user can check what procedures he has assigned up till that time, by selecting the Added Procedures tab.
    - After the user finishes assigning procedures, he can finish the assignment process by clicking on the button Finish Product. After that he can see what products have been finished by selecting the Finished Products tab of the list. When the user has finished at least 1 product and has filled the Material Price and Year-Month-Day fields, the button Confirm will light up and the user can finish the creating offer process, which will result in a new offer added in the system and the user will be returned back to the Offer Menu.
    -- In case the user wants to add more than 1 product to the offer, he can simply

repeat all the steps above and he easily checks the added products up till that moment from the Finished Products tab of the list.

-- A total price for the whole offer will automatically calculated and generated based on the number of products, procedures and the price of the material the user has inputted. He can easily see how much is it by returning to the Offer Menu and look at the list of offers.

1.3) Update Offer Menu:

- All the current information about the selected offer will be displayed on the screen.

- If the user wants to change part or all of the information, he edits the input fields however he likes.

- If the user has made changes and wants to confirm the update, he clicks on the button Confirm.

- If the user doesn't want to update anything, he clicks on the Cancel button and he will be returned back to the Offer Menu.

1.4) Delete Offer Menu:

- The user will be prompted whether he is sure he wants to delete the selected offer.

- If he wants to delete it, he clicks the button Yes.

- If he doesn't want to delete it, he clicks on the button No.

1.5) Manage Procedures Menu:

- This is the menu where the user can create, update, delete, search and view all currently existing procedures in the system. For help check steps 1.1 - 1.4, since they share similar designs.

2) **Sale Order:**

2.1) Sale Order Menu:

- Please refer to step 1.1 for information

2.2) Create Sale Order Menu:

- If the user wants to create new sale order, he does the following:

2.2.1) Choose Offer Menu:

- The user selects an offer from the list of offers and presses the button Continue in order to proceed to creating a new sale order.

- In case there are no offers displayed, the user must go back by clicking the button Go Back and then navigate to Offer Menu in order to first create an offer before returning back to the creating sale order process. For further reference about creating an offer, check step 1.2.

2.2.2) Create Sale Order Menu Dialog:

- A new window with a list of products, several input fields and 2 selection boxes will appear. The user will notice that the offer number, due date, order date, total price will automatically be filled, which are taken from the offer number they previously chose.

- The user still needs to fill out the remaining fields before confirming the creation of the sale order.

- The user can only choose customers that exist in the system from the customer

selection box.

- The user can only choose products that the user previously added to the offer when the offer was being created. After selecting a product, the user will notice that the quantity input field will be automatically filled, which is again taken from the offer. User still needs to fill out description and price of the selected product. After having done all previous tasks, the user can proceed to clicking the Add Product button, which will add the selected product to the list of products that is displayed in the middle of the screen.

-- In case the user made a mistake while adding a product to the list, he can easily remove it from there by selecting it on the list and clicking the Remove Product button.

- Previous steps can be repeated many times until all desired products have been added to the list.

- By clicking the Confirm button, the creation process of the sale order will be finalized, the sale order will be created and added to the system. The user will be returned back to Sale Order Menu.

2.3) Update Sale Order Menu:

- Please refer to step 1.3 for information

2.4) Delete Sale Order Menu:

- Please refer to step 1.4 for information.

3) **Customer**

3.1) Customer Menu:

- Overview of the existing customers and control over managing them. For further information check 1.1.

3.2) Create Customer Menu:

- When the user presses this button, a small window pops up and requires some information to be inputted before the user can finish the creation of the customer.

3.3) Update Customer Menu:

- Allows the user to update the information of a customer. For help check 1.3.

3.4) Delete Customer Menu:

- Allows the user to remove a customer from the system. For help check 1.4.

4) **Employee**

4.1) Employee Menu:

- Check 3.1 for information.

4.2) Create Employee Menu:

- Check 3.2 for information

4.3) Update Employee Menu:

- Check 3.3 for information

4.4) Delete Employee Menu:

-Check 3.4 for information

4.5) Manage Positions Menu:

- This menu is used for creating, updating, deleting, searching and getting a view of all positions currently existing in the system. For more information about how to do these actions, check 3.1 - 3.4, since they share similar designs.

5) **Scheduler**

- The scheduler menu is used for getting information about which employee is currently working on which procedure.
- After clicking on the button Scheduler, a small window will appear, which consists of only one list and a button Done.
- The list in the middle of the screen can be used for viewing all currently assigned and working employees on procedures.
- If the user wants to mark an employee, whether he is done or not, he can achieve that by first selecting an employee from the list of employees and then clicking on the Done button. This will cause the window to automatically refresh and be refilled with the latest information regarding employees and procedures being worked on.

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
Sebastian Labuda: Support

Carry: basic

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 usable.

Tank:

Tank should create a structure of the project. Catching errors and misses, they provide stability to the group's work, thus protecting 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 provides 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.

Group work evaluation

As can be seen in our group contract, the purpose of our group is to create a unique environment, where learning is our first priority. By creating our group contract, we clearly set rules that should be respected by every single member, as well as setting roles, tasks and responsibilities for certain parts of our project.

Second semester proceeded in accordance with the current pandemic situation around the world. Whole pandemic situation quickly spread around the world, influencing the lives of millions of people. Our group was not different. Our group had to face multiple new problems and situations, which we haven't faced before. We had to quickly adapt to changes regarding our studying as well as normal way of living. Online classes alongside as well as working on the project proved very challenging. Hardest part of this project was this time system development. Because of the current situation, setting up a proper logic on which all of us will agree proved very challenging through the voice calls. For a few days, misunderstandings were on a daily basis. There were no disputes, but we simply understood things differently, which created conflicts. We eventually found a way to work in this “new” environment, but it took some time.

On the other hand, we stepped towards the whole situation with maximal respect and understanding. Regardless of our negative attitude (at the beginning) and overall disposition, every member gave it best on working on the project. Each member of the group worked the same way, including any part of our project. At the end, we find in a pandemic situation a great opportunity. Situation served as a great test. We had a chance to use all our knowledge and studying materials we have learned through both semester and even increase our knowledge, as well as to test

Conclusion

Overall we have met the requirements and have reached our goal. We have implemented the two most complex use cases and we have the foundation for the other three. Especially our second use case - the scheduler, was something that we had never encountered before. It uses lambdas, loops and threads extensively and the whole logic of scheduling hides itself behind an ordered graph.

Even though we have implemented a huge part of the system, the project is not finished. Besides finishing the rest of the use cases, one of the things that could be done is to make the GUI even more user-friendly. Having a help window in each part, which would describe a certain part of GUI could help the users a lot. To ensure its quality, testing this GUI with users and taking into consideration their suggestions could improve the quality even more. Other improvement could be adding even more checks to completely ensure that users cannot in any circumstances break the program.

This was our first big project in terms of time and complexity, so there was a lot we have learned. We have become better in planning and developing complex projects. Considering the pandemic situation, we have learned how to cooperate better, since more problems and conflicts arose

which we had to conquer. It also proved the importance of proper planning and risk analysis, where taking it lightly could jeopardize our whole project.

Literature

[1] wikipedia.org. [Online].
Available: https://en.wikipedia.org/wiki/Unified_Process
[Accessed: 18-May-2020].

[2] crn.com.au [Online].

Available: https://www.crn.com.au/news/bankrupt-reseller-grew-too-much-too-fast-344930

[Accessed: 23-April-2020].

[3] transparentchoice.com [Online].

Available: https://www.transparentchoice.com/strategic-alignment

[Accessed: 27-April-2020].

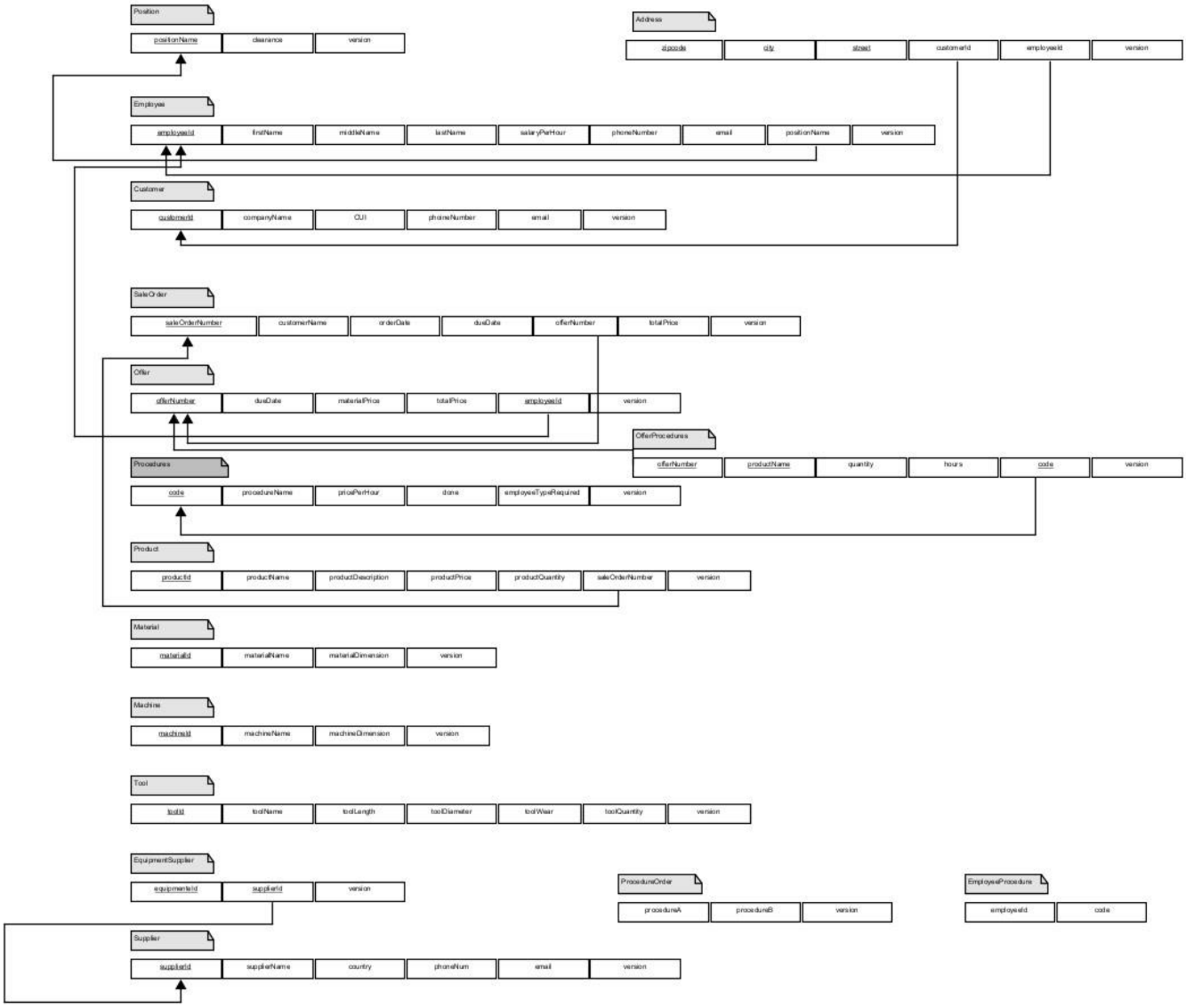
[4] Oracle.com, 2019. [Online].

Available: https://www.oracle.com/technetwork/java/codeconventions-150003.pdf.

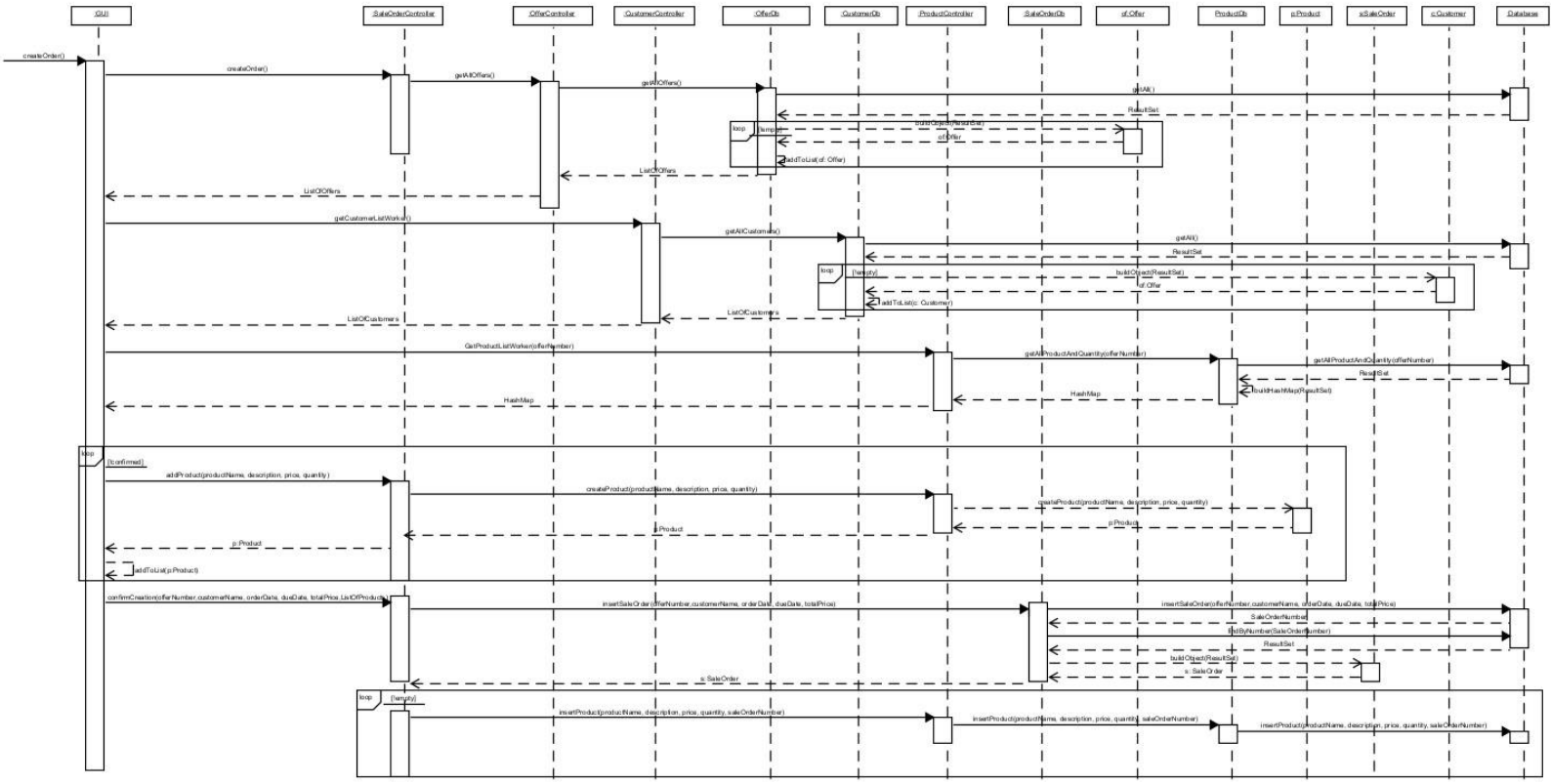
[Accessed: 26-Mar-2020].

Appendices

Appendix 1 Relational model



Appendix 2 Interaction diagram



Appendix 3 Design class diagram