ChemStruct

Gerald Mahlknecht

TFO “Max Valier “ Bozen

2016 – 2017

Tutors: Prof. Michael Wild & Prof. Alexander Coccia

Contents

List of Figures

1. Introduction

1.1 Chemical Background

Each substance consists of molecules, which are built from atoms and their bounds. A chemical structural formula is one way to represent these chemical compounds in a chart. It shows the molecular structure formed by the arrangement of the atoms. Each structural formula has a special name, which can be identified with chemical nomenclature. There are different normed notation modes. The most used are the "Lewis structures", the "Condensed formulas" and the "Skeletal formulas". The first two notations are mostly used for small molecules. For my project, I used "Skeletal formulas". From the huge number of different molecules in the first version of my program I focused on the hydrocarbons, which are part of the organic chemistry. These molecules consist only of carbon (C) and hydrogen (H) atoms. They can be divided in two classes, i.e. chain configurations and cyclic configurations. A C atom always connects with four other atoms. The number of the single bounds depends on the other atom and on the number of not used valence electrons. Between two C atoms can be a single bound, a double bound or a triple bound. All free valence electrons are bound with H atoms, which have only one valence electron.

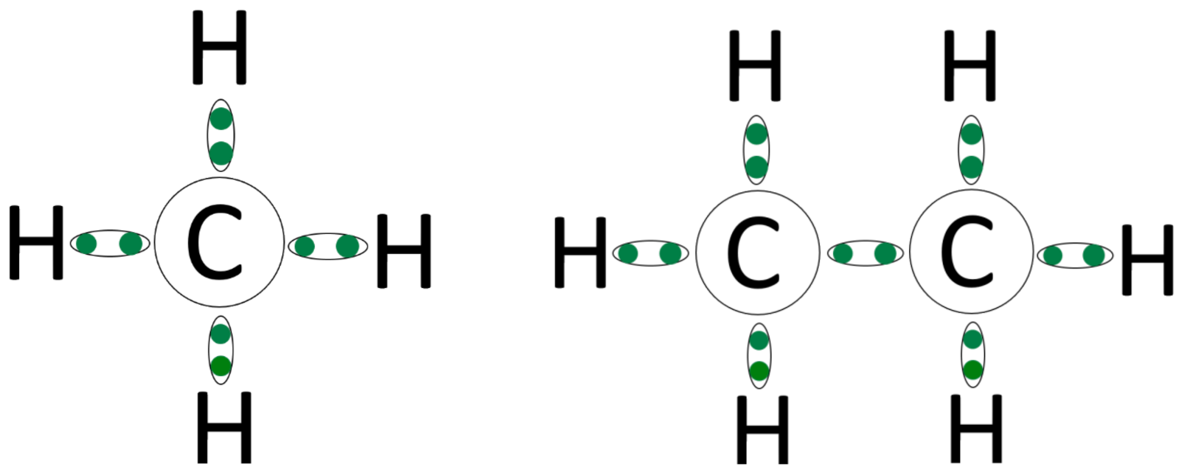


Figure 1: Example of C and H atom bounds

1.1.1 Skeletal Formula

Skeletal formulas are a graphical representation of molecules. They only have two dimensions and do not show valence electrons and other atoms than C atoms. Other atoms are represented by element symbols and the bounds by a solid line. Double bonds are shown with two lines, triple bonds with three lines.

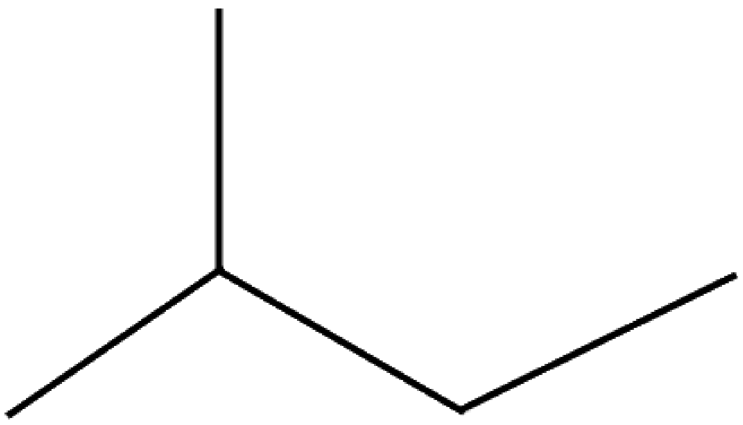


Figure 2: Skeletal formula of 2-methylbutane

1.2 The Problem

Studying chemistry at school I had problems in understanding my own drawings of hydrocarbon structural formulas. Most of the time I ended up searching for some formulas on the internet that could be the ones I needed. I had difficulty in finding the right name and I’m sure so did many other students too.

1.3 The Idea

1.3.1 Birth of the Idea

Although I do not have chemical classes any longer, I sometimes like to look up the name of a specific hydrocarbon structural formula. I tried out different software but with limited subject-specific knowledge it was a hard process to get the result I wanted. That was the point when I decided to create my own program to name simple structural formulas for hydrocarbons with just a few clicks.

1.3.2 The Target Group

It is quite simple to draw, save and name hydrocarbon chemical structure formulas with my software. This means the whole system is based on non-complex structural formulas. The target group is pupils in their first two years of chemistry lessons. For school purposes the notation “Skeletal formulas” is most frequently used. That is why I decided to focus on them.

1.4 The Motivation

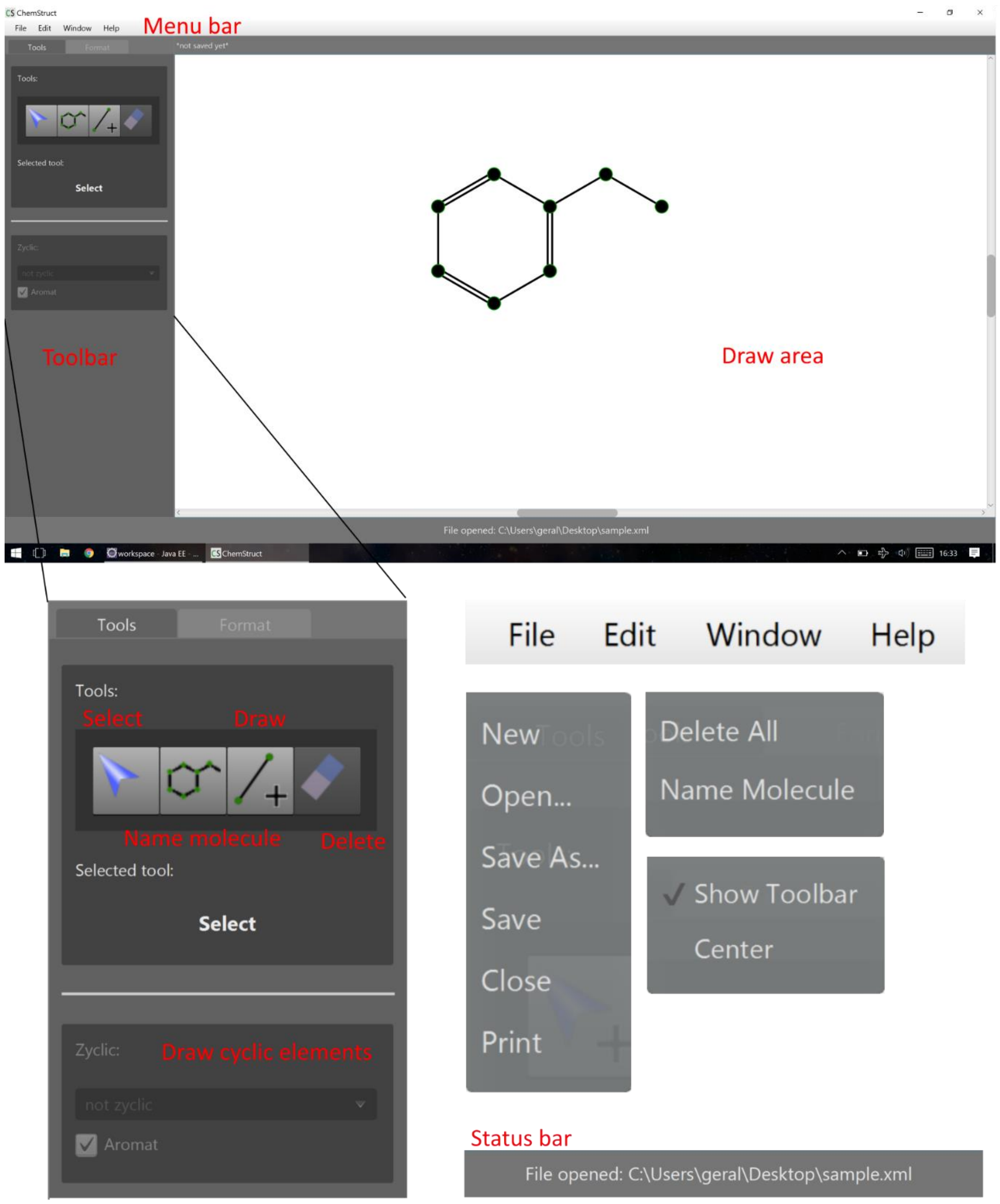
I got motivated by Coccia Alexander, my former chemistry teacher. he was looking for someone to develop a software for him and his students that makes it possible to draw chemical structural formulas on a blank page, to name them afterwards and to print them for laboratory reports or to use them as a learning tool. There are some alternative applications that have similar functions but none of them focuses on the basics or is even easy to understand for a non-professional person.

1.5 The Goal

I wanted to create a software that makes the process of drawing and naming structural formulas fast and simple. As a result, I wanted to provide a system with a “learning-by-doing-effect”. Therefore, I created a window application named “ChemStruct”.

2. ChemStruct

2.1 User Interface



File opened: C:\Users\geral\Desktop\sample.xml

Figure 3: Start layout of ChemStruct

After starting the application, the user interface appears. On the right-hand side of the screen is the core element, which is the drawing area. To its left is the toolbar with the most important editing tools. Underneath the tools is a status bar which shows the currently selected tool. By switching tool, the status bar also changes. The format tab is enabled under different selection conditions and provides object-specific functions such as changing the bound size. The menu bar contains the second level operations like reset, save options etc.

2.1.1 Why the Dark Design?

I spent a lot of time improving the user interface. My aim was to make it look pleasant and manageable. The user should not get the first impression of a vast and complex application. I realised that not only does the array of the components make a good effect but also the right mixture of colours. I was inspired by dark-shaded developing environments like Eclipse and Android Studio. The colour management is the result of a mixture of different shades of grey.

2.2 Use Cases

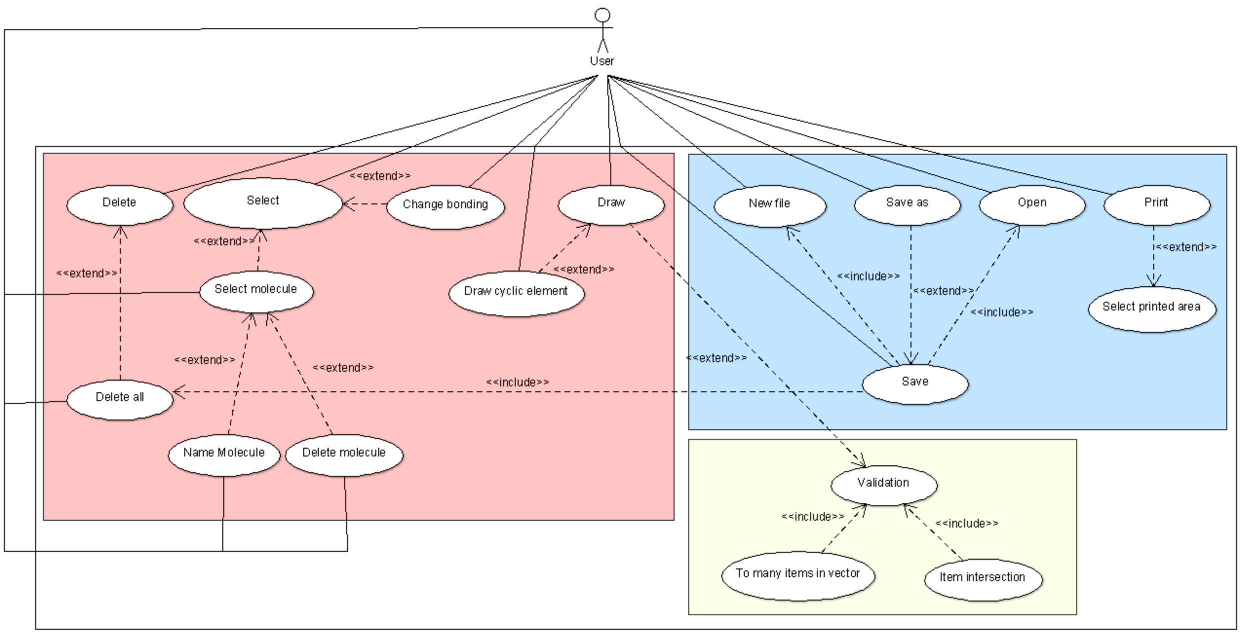


Figure 4: Use case diagram

2.2.1 Draw

One of the core features of ChemStruct is the drawing process of molecules. Once the user started the program he/she is in front of a wide drawing area. Simply by selecting the “drawing-tool” and clicking in this drawing area, it is possible to start designing a molecule. He/she can extend it to another bound by clicking on the elliptical objects, which represent C atoms. Just clicking on them will manually set a default angle for the new bound. By dragging the mouse to the desired position after clicking on the requested atom-object, a new bound is created with this angle. Each C atom can have a maximum of 4 bounds. The manually calculated angle depends on the number of bounds at the moment of adding a new one and of the angle of these bounds.

2.2.2 Delete

A context-menu is shown over the selected object after right-clicking on it. Depending on the type of the object it provides different operations like the “delete-option”, which is implemented for all kinds of drawing components. The problem at this point is that each component is bound to others and could be a central branch. I created an algorithm that handles the deleting process and decides which parts have to be deleted and which do not, according to the given situation.

2.2.3 Change Bounding

A molecule can be formed by single, by double and by triple bounds. Therefore, the context-menu for bounds has a sub-menu to select the kind of bound. With regard to the number of valence electrons the maximum bound type is a triple bound.

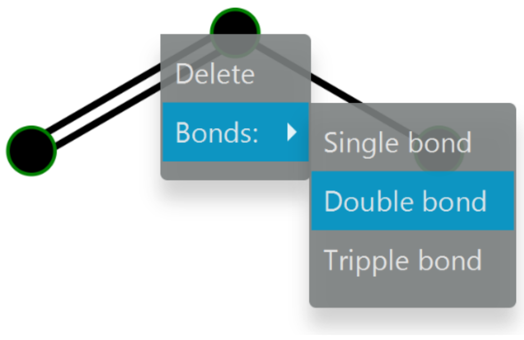


Figure 5: Context menu for bounds

2.2.4 Cyclic Components

As an important part of molecules cyclic components must not be excluded. To optimize their drawing process the “drawing-tool” supplies a range of pre-built cyclic molecule components. By selecting one of the given options from cyclo propane to cyclo octane, they can be created in the drawing area like normal bounds. In this case, the selected angle defines the angle of the first bound. A sub-option of cyclic components is the “aromat-function”, which means, that each second bound of the cyclic object gets automatically equipped with a double bound.

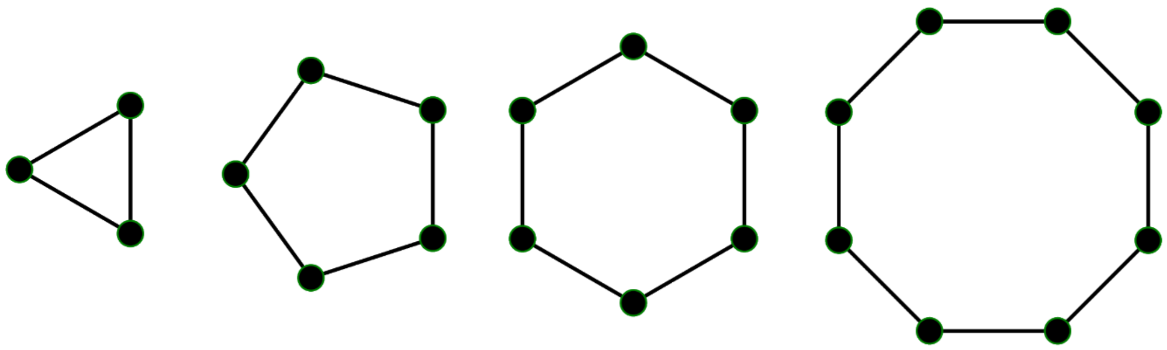


Figure 6: Example of possible cyclic components

2.2.5 New File

“New file” asks the user in case of unsaved content in the drawing area if he/she wants to save all current changes. After that, the reference to the persistent instance of the file is overwritten with an empty file and thus the drawing area is cleared.

2.2.6 Save (As)

To handle the save option I decided to design the function following other popular programs. After pressing the save-button the system checks if a saved instance exists for the current opened file. If there is a saved instance, the file gets overwritten, if there is not, a “save as” dialog appears and the preferred file direction can be selected. This flow accelerates the process of saving and provides the possibility to save more than one instance of the object. All files are saved in standard XML-Format.

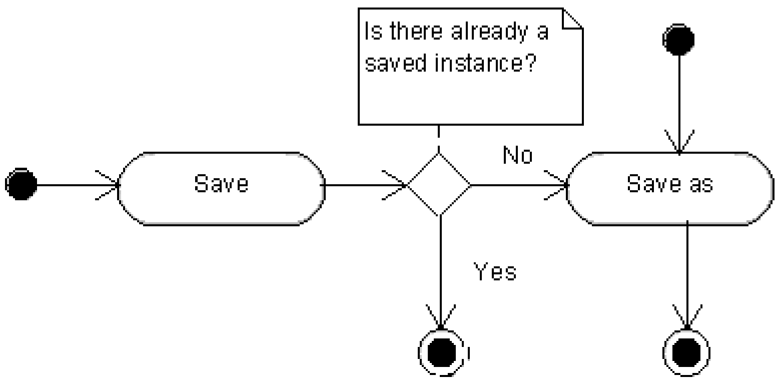


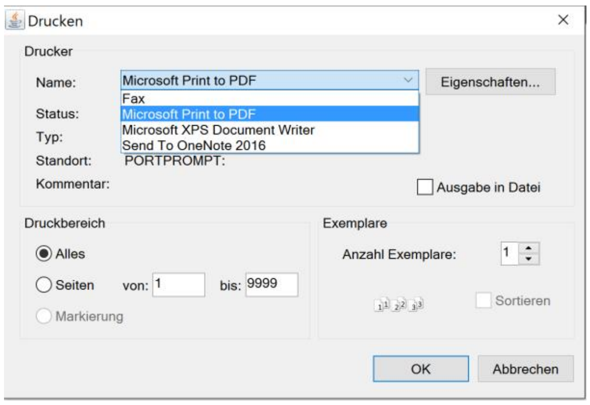
Figure 7: Activity diagram of the save process

2.2.7 Open

Equal to the “new file” operation the “open” operation asks the user if he/she would like to save his/her changes if there are any. The saving process is the same as mentioned before. Afterwards a file chooser enables to select the file to be opened and its content is displayed in the drawing area. In order to avoid confusion, I added an XML-file-filter for the saving and opening process.

2.2.8 Print

Due to the big size of the drawing area, which the user might not want to print as a whole, I created a selection tool to select the desired content. A hint shows up in the error and status field. Because of the necessity of a printable file which is not written in XML, I used a choosing tool that provides a PDF-print function.


Drucken

Drucker

Name: Microsoft Print to PDF — Eigenschaften...

Fax
Microsoft Print to PDF
Microsoft XPS Document Writer
Send To OneNote 2016

Status:
Typ:
Standort: PORTPROMPT:
Kommentar:

☐ Ausgabe in Datei

Druckbereich

● Alles
○ Seiten von: 1 bis: 9999
○ Markierung

Exemplare

Anzahl Exemplare: 1

☐ Sortieren

OK — Abbrechen

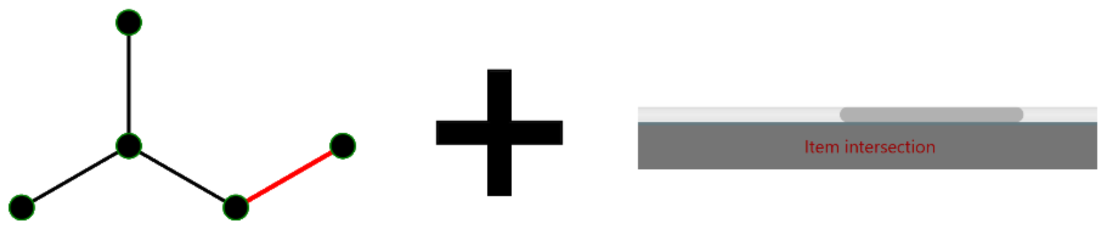
Figure 8: Printerjob print dialog

2.2.9 Validation and Error Messages

There is a runtime validation of the drawing area that denies intersections of atoms and bounds and several other possible occurring errors.

2.2.10 What Happens if an Error Occurs?

To ensure a continuous working flow the system catches occurring exceptions and handles them. If these errors are created by the user like in the case of an intersection or an atom with too many bounds, the system reacts with an error message that shows up below the drawing area and sets the colour of the affected element to red. Each created component gets verified.

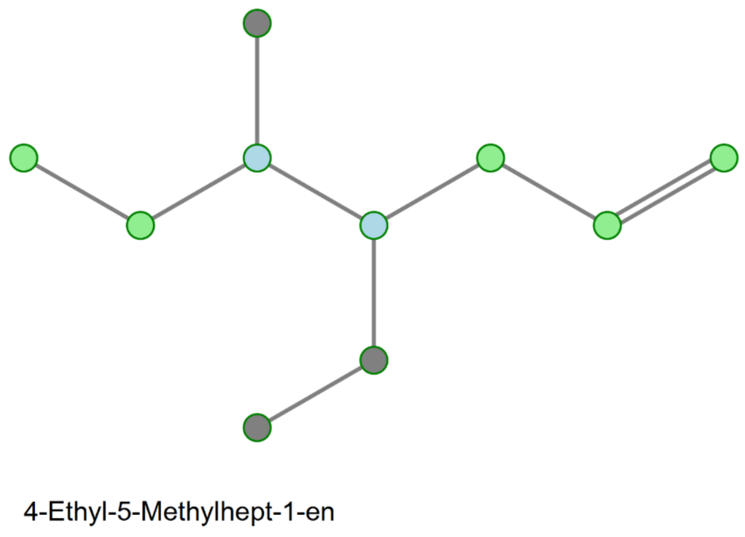


Item intersection

Figure 9: Visualization of user made errors

2.2.11 Nomenclature

The nomenclature can be made by selecting a molecule with the “molecule tool” or with the function in the menu bar. Its name appears underneath the selected object. This name disappears after re-editing the molecule to avoid inconsistency.



4-Ethyl-5-Methylhept-1-en

Figure 10: Example of a named molecule

3. Get Technical

For the drawing as the main and most complex process of the application I want to give a more detailed overview of the technical implementation. To make the component understandable it is divided up into its chronological sub-processes.

In reality a molecule is made of atoms and their bounds. I tried to implement this fact in my system as closely as possible to reality. There are objects for atoms and object for bounds. An atom knows its bounded atoms through the bound. There is no direct reference between atoms. The positive side-effect of this is flexibility. The process of handling the number of atoms bounded gets adaptable due to the fact that each atom can be bound to a fixed number of other atoms according to its element. Not only can atoms but also bounds and their co-existing bounds in a multiple bound be handled more easily if they are separated.

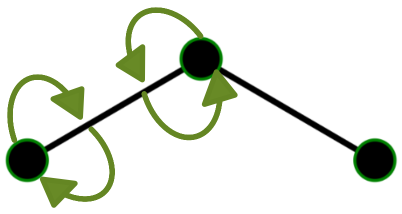
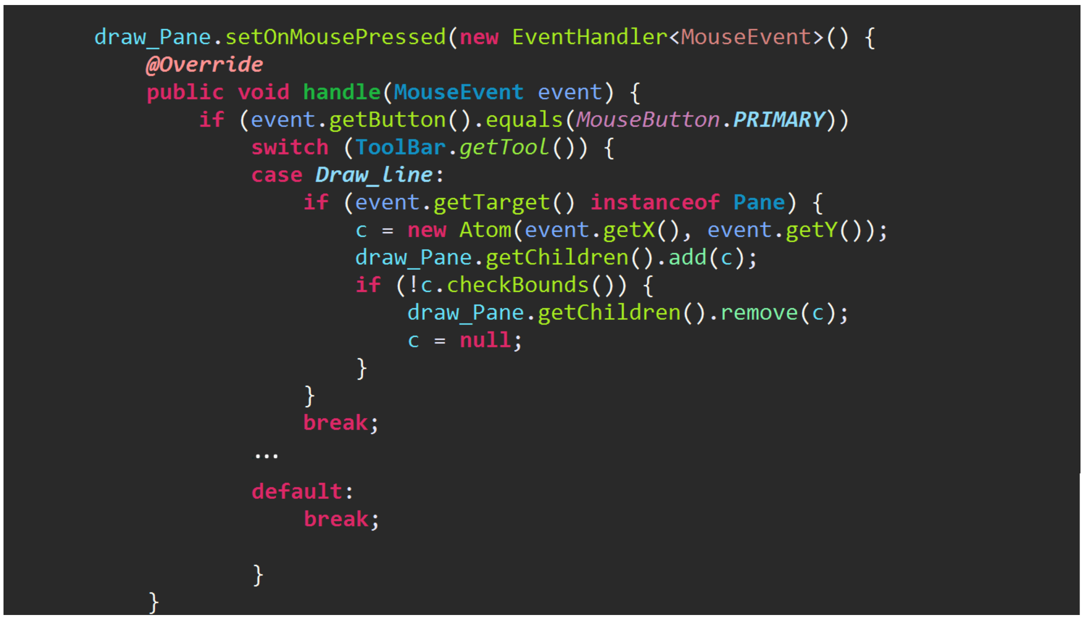


Figure 11: C atoms and their references

3.1 Atom

Everything starts with a single atom. A mouse-pressed-listener detects the x- and y-coordinates in the drawing pane and creates a new atom object with the coordinates in its centre. Atom extends Circle (javafx.scene.shape.Circle), which is defined by layout coordinates and a radius. It is important to point out that at this point this object has no bounds yet. Now the first validation begins. Atoms as well as bounds have a method called “checkBounds” which checks if the object intersects other objects by parsing all objects in the pane. If the method returns “false” the new atom object is removed from the pane and set on null to avoid further complications. At this point the system throws a “DrawException”, which is displayed on the status label.



```
draw_Pane.setOnMousePressed(new EventHandler<MouseEvent>() {
    @Override
    public void handle(MouseEvent event) {
        if (event.getButton().equals(MouseButton.PRIMARY))
            switch (ToolBar.getTool()) {
            case Draw_Line:
                if (event.getTarget() instanceof Pane) {
                    c = new Atom(event.getX(), event.getY());
                    draw_Pane.getChildren().add(c);
                    if (!c.checkBounds()) {
                        draw_Pane.getChildren().remove(c);
                        c = null;
                    }
                }
                break;
            ...
            default:
                break;

            }
    }
```

Figure 12: Inizialisation of an atom object

3.2 Bound

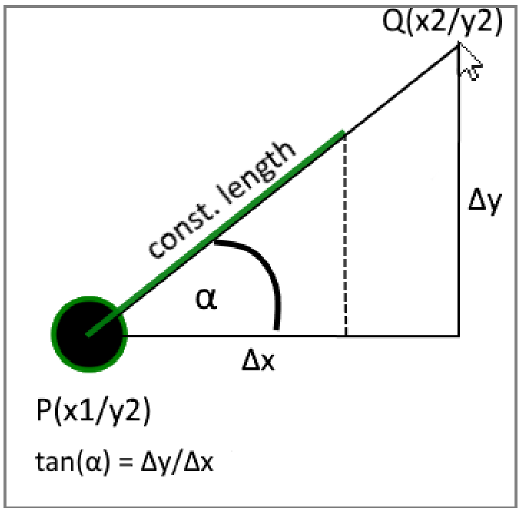


Figure 13: Calculation of the bounds angle

If the validation method returns "true" the atom object is still set. A mouse-release-listener now registers once again the mouse coordinates at the moment of releasing. The method "addLine" is called on the atom object with the new coordinates as parameters. This method creates a new bound object and handles the linkage of the atom and the bound. The bound class extends Line (javafx.scene.shape.Line). Lines and their extended objects are a bit different in handling compared to other JavaFX shapes. They do not only have one point of reference with x- and y-coordinates but one start and one end point. Compared to the new coordinates and the coordinates of the object on which the method is called the angle gets calculated and set. Finally, it is added to the pane. As on the atom object, now on the bound object the "check-Bounds"-method is called. If it returns "false" the bound and its atom are removed from the pane and an exception is thrown. If there is no item intersecting but the bound's own atom, the bound tries to add another atom at its second ending. The process is the same as described in "addLine". This step-by-step-validation makes it possible to stop the drawing process at the point when the error occurs and not after running the whole process.

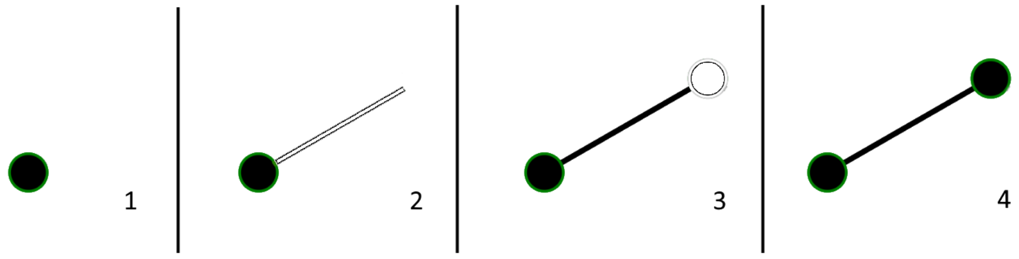


Figure 14: Step-by-step creation of an atom bound

3.3 Increment and Decrement Bounds

To change bound size, I did not really change anything but created or deleted bounds. The original approach was to just change the style of the bound object which was dismissed due to problems of missing style attributes in the superior object Line. If two atoms are bound through a multiple bound, the individual lines could exist without each other. They do not differ in their style or their core attributes. Each multiple bound has a main-line with listeners. All other lines have a reference of this line and in case of a mouse event it is consumed and forwarded to its main-line.

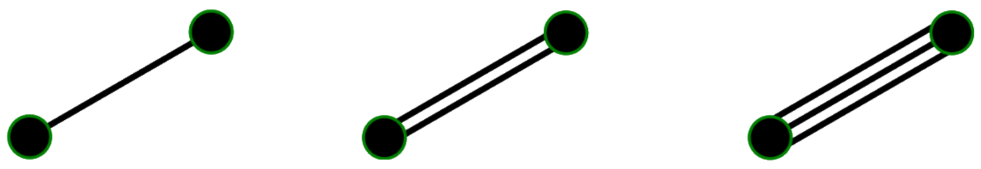


Figure 15: Possible multiple bounds

3.4 StateHandler – Current State

I tried to outsource everything possible to the single objects. That is why it is important that each of them knows which tool in the toolbar is selected. I created a static state handler which is controlled by the tool bar and is accessible for all components without initialising it. Controlled by the tool bar in this case means that the tool bar changes the current state. My intention was to avoid errors with wrong states. Therefore, I created an enumeration containing the possible values.

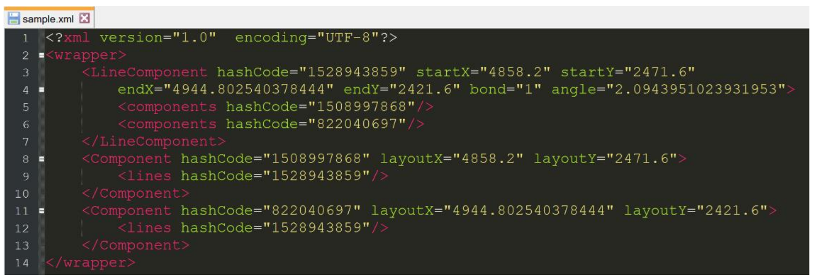
3.5 DrawException

Particularly for errors made by the user I developed a type of exception. It extends java.lang.Exception. For reasons of consistency I left the functionality as it originally was. For the output function in the status bar I added an optional Label which can be set over a setter method. Only if the Label is set, is the exception message written in the status bar. At this point I spent a lot of time to make the user aware of obscurities. Once an error occurred, it was displayed in the status bar, but it did not disappear until the next error occurred. Therefore, every time something goes wrong I initialise a timer[1], which hides the status bar after a set time.

[1] (Oracle, Oracle 2015)

3.6 Saving and Opening in XML

One of the last implemented features was the “saving” and “opening” operations. I already mentioned the work flow so now I would like to explain the functionality. The objects in the drawing pane are listed as a non-standalone tag with the tag names “LineComponent” for bounds and “Component” for atoms. Its attributes are its hash code, coordinates, in case of a bound object start and end coordinates, if existent the angle and the bound size in case of bounds. The sub elements are standalone tags and represent bounds and/or atoms corresponding to the parent tag. They are only defined through their hash code. The sub-elements are necessary to rebuild the original connections among the saved elements. By calling the save method all components in the drawing pane create their own XML code section thanks to the “toXML” method.



```xml
<?xml version="1.0"  encoding="UTF-8"?>
<wrapper>
    <LineComponent hashCode="1528943859" startX="4858.2" startY="2471.6"
        endX="4944.802540378444" endY="2421.6" bond="1" angle="2.0943951023931953">
        <components hashCode="1508997868"/>
        <components hashCode="822040697"/>
    </LineComponent>
    <Component hashCode="1508997868" layoutX="4858.2" layoutY="2471.6">
        <lines hashCode="1528943859"/>
    </Component>
    <Component hashCode="822040697" layoutX="4944.802540378444" layoutY="2421.6">
        <lines hashCode="1528943859"/>
    </Component>
</wrapper>
```

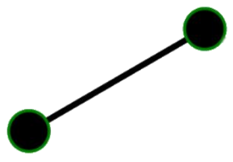


Figure 16: XML code for a simple bound

During the “opening” process the file is parsed with an XML parser. The parser is created on a base frame put at disposal in raw data by the information technology “Metakurs”[2]. The parsing process is repeated twice. In the first turn only the objects are set with their properties, which are saved in the attributes and afterwards added to the drawing pane. During the second turn the parent elements get their sub-elements set. Due to the fact that all items now exist in the system, they can easily be connected by using their unique hash code.

Both, “saving” and “opening” are outsourced to a class called “DocumentHandler”. This class also has the function to keep the reference to a saved file if there is one.

3.7 Why not XStream or JAXB

Both XStream[3] and JAXB[4] are libraries that provide features to marshal objects to XML and unmarshal them back. After many attempts, I found out that those libraries,

[2] (Wild, eXtensible Markup Language (XML) kein Datum)
[3] (Jakob, code.makery 2013)
[4] (Jakob, code.makery 2015)

even if there are tutorials particularly for the usage in JavaFX, are not compatible with objects which extend JavaFX Nodes.

3.8 Preferences

I have seen different tutorials in which the path is saved with a class called “Preferences”. Preferences[5] are saved at different places depending on the operating system. Those saving locations have a limited amount of memory which is relatively small, but big enough to contain an absolute path.

3.8.1 Why not Preferences?

I decided to realise the caching of the file’s reference with a simple string containing the absolute file path of a saved instance. I did this, simply because I found out that depending on the computer running the program, an error could occur by using Preference. The solution would be an added entry in regedit. Due to the fact that users do not have a development console they could not even look up the error and would be surprised about this bug.

3.9 Algorithms of Nomenclature

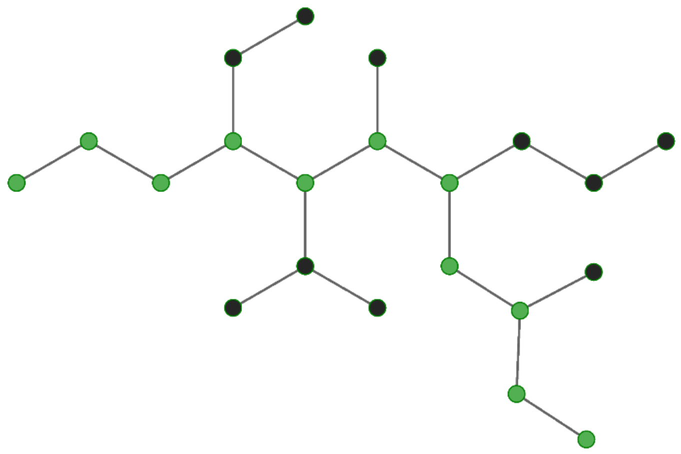


Figure 17: A*-algorithm for the longest path

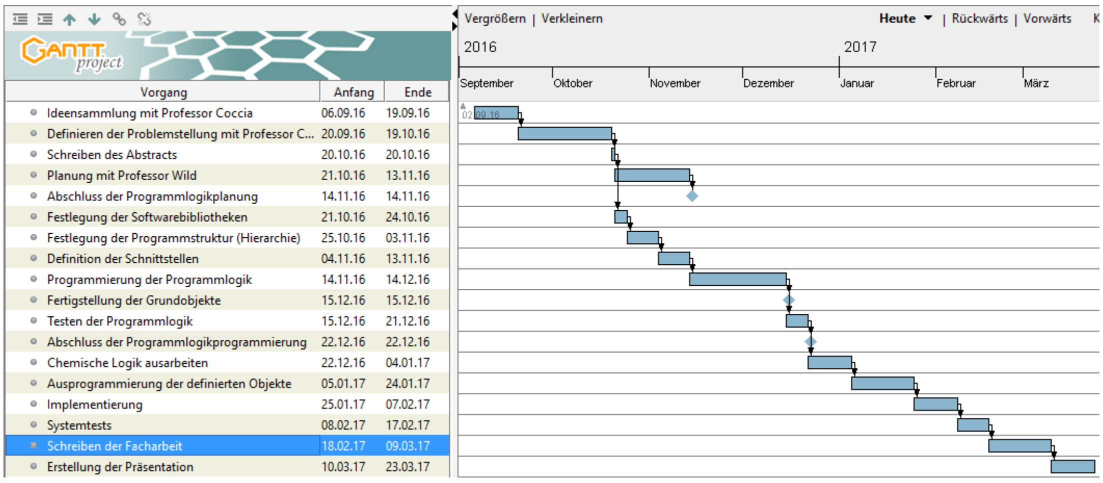
To name a molecule first of all a molecule has to be selected. To save memory not all molecules are cached. The implemented molecule object can make one molecule reference at a time. If it is unset its reference is on null. By clicking on an atom with the “molecule” tool the molecule starts creating itself by calling several recursive methods. To follow the nomenclature standards the first step is to search the longest

[5] (O. Unknown, Oracle 2016)

chain. I implemented this with a modified A*-algorithm with the same metric for all knots. In the best case this method returns two paths with the same length, which both represent the same path only mirror-inverted. There are cases in which two different paths with the same length are detected. For these exceptions, I needed to implement a second method to filter out the right one. The right one means the one with the longest side chain has a higher priority. The now filtered out longest path has two possible beginnings. Once again in a recursive method starting from the two start points until reaching the end of this chain, all side chains and their position are detected. The algorithm finds out from which side the nomenclature would result smaller numbers. Following the nomenclature rules at this point a last algorithm creates the final name out of all detected side chains, their position and the longest chain. The single components are set in alphabetic order and presented appending an offset underneath the atom object with the biggest y-coordinate.

4. Schedule and Development

The ChemStruct-project is my first complex project. Unfortunately, I could not stay on schedule due to some wrong assessments.



Vorgang	Anfang	Ende
Ideensammlung mit Professor Coccia	06.09.16	19.09.16
Definieren der Problemstellung mit Professor C...	20.09.16	19.10.16
Schreiben des Abstracts	20.10.16	20.10.16
Planung mit Professor Wild	21.10.16	13.11.16
Abschluss der Programmlogikplanung	14.11.16	14.11.16
Festlegung der Softwarebibliotheken	21.10.16	24.10.16
Festlegung der Programmstruktur (Hierarchie)	25.10.16	03.11.16
Definition der Schnittstellen	04.11.16	13.11.16
Programmierung der Programmlogik	14.11.16	14.12.16
Fertigstellung der Grundobjekte	15.12.16	15.12.16
Testen der Programmlogik	15.12.16	21.12.16
Abschluss der Programmlogikprogrammierung	22.12.16	22.12.16
Chemische Logik ausarbeiten	22.12.16	04.01.17
Ausprogrammierung der definierten Objekte	05.01.17	24.01.17
Implementierung	25.01.17	07.02.17
Systemtests	08.02.17	17.02.17
Schreiben der Facharbeit	18.02.17	09.03.17
Erstellung der Präsentation	10.03.17	23.03.17

Figure 18: Gantt diagram of the program schedule

To show all important objects I created a business object model. The most important parts of it are the “Atom” and the “Linkage” objects. Both extend “SuperComponent”, a superior object to make conversions easier to handle. “Atom” and “Linkage” have a direct reference and a composition. This means a “Linkage” has to have two “Atoms”. Another core object is “Molecule” which consists of a flexible number “Atoms” and “Linkages”. Due to the fact that a molecule exists no longer if it changes, there is a composition too. Except the “DrawException” and the “MainController” all other classes are static file or data handling classes. For example, the “Factory” class is a static class which creates cyclic components.

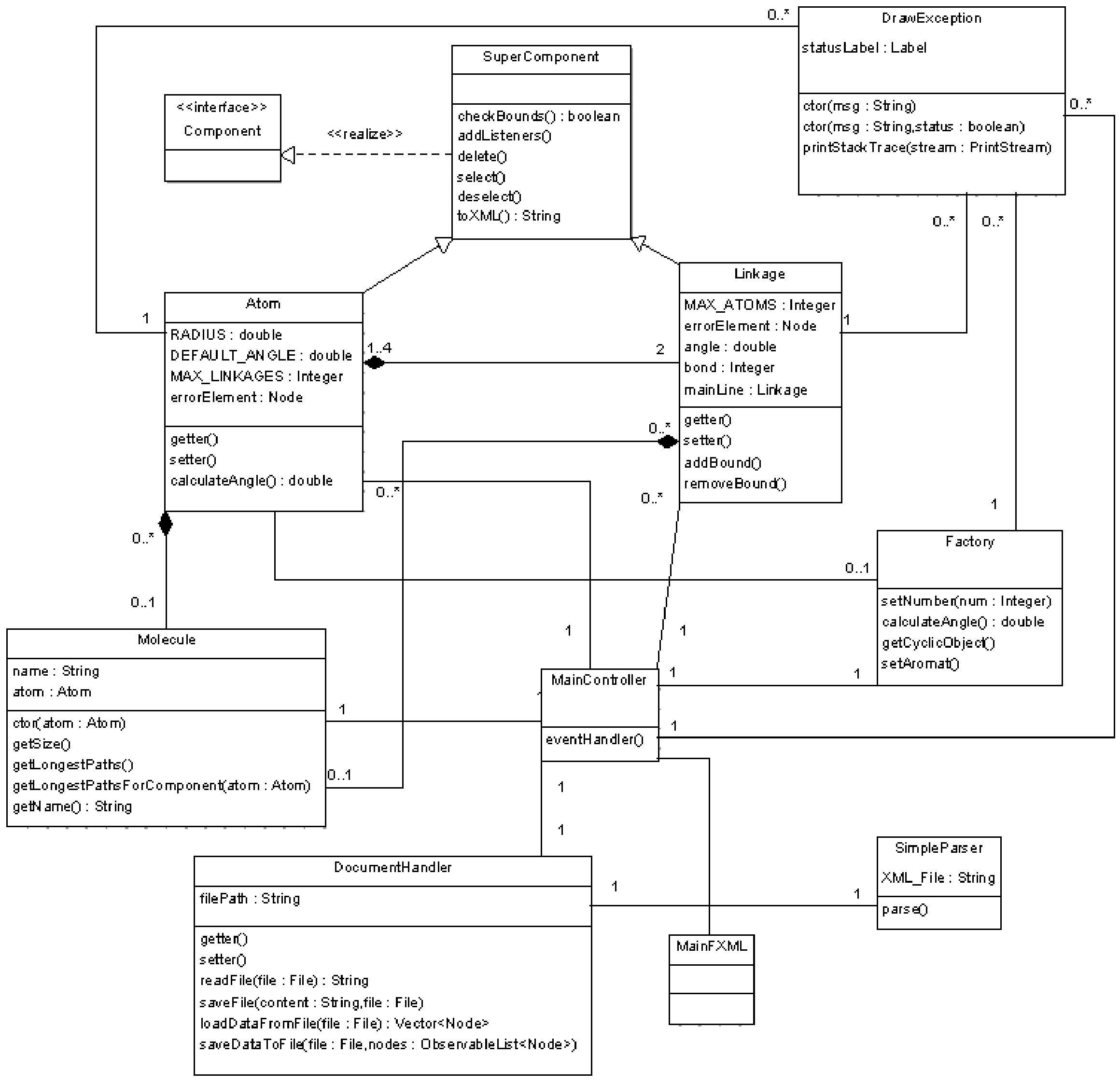


Figure 19: Business Object Model

4.1 Spiral Model

As it is evident from the Gantt diagram above I developed this project with the spiral model. The spiral model is a process model for software projects and is an advancement of the waterfall model. It divides the main planning and developing process into iterations. Each iteration is made up of four sub processes: fixing goals, detecting and resolving risks, development and testing and planning the next iteration.

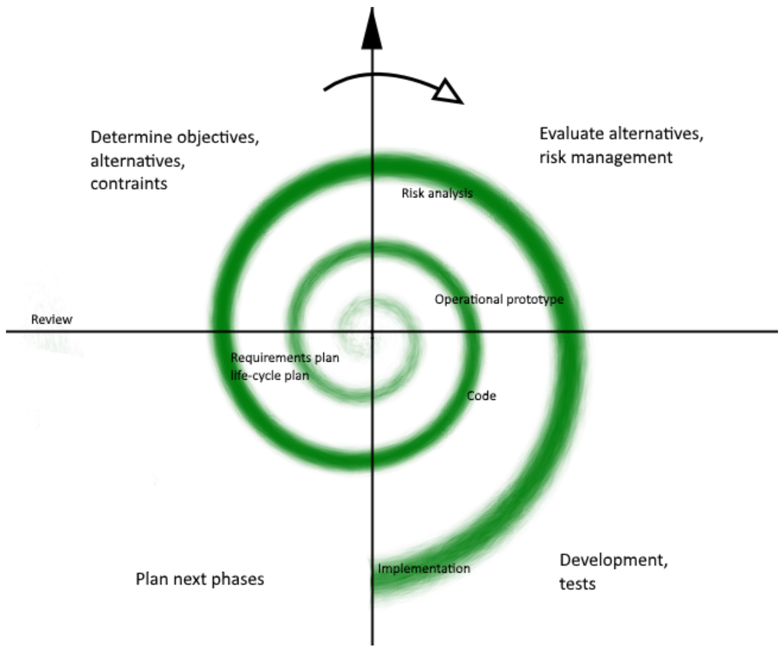


Figure 20: Spiral model

4.1.1 Why the Spiral Model?

The advantage of this development model is the step-by-step development. The focus lies on the risk management and qualifies this model for complex projects. The main reason I selected this kind of model is to have the possibility to add functionalities one by one and have a base on which I can later work on. I will explain in the following paragraphs how I worked with the model:

4.1.2 Fixing Goals

At the time I designed the Gantt project, I divided up the main problem into two big sub problems, the drawing logic and the program logic. However, just after a short period of time I realised that each of them had to be divided up into really small fragments and that there also had to be background algorithms for the molecular logic. I began to define the goals for the drawing process. First of all, a single linkage of two atoms and their bound in a fixed angle should be constructible. In the next two iterations, I set the goals on the implementation of the connection among the objects and on a method that can calculate a changeable angle and the angle of a third bound. I fixed goals for the editing processes all in one iteration except for the bound change because the effort was comparatively small to the other iterations. After the following iteration with the changeability of the bounds and the development of it I continued with the program logic. My first goals were the development of algorithms for the molecular logic. The following aims were a rough draft of the layout and the functionality of the toolbar. The next-to-last goal was a complete functional interface with a smart design. To round off the work flow I also fixed goals for shortcuts and context menus. According to the planning process I had different possible ways for the final implementation and before I started working on it, I did some research on alternatives or already existing libraries. An example of this was the “PrinterJob”-library[6], which saved me hours of time.

4.1.3 Detecting and Resolving Risks

I worked with a series of software components which were unknown to me and therefore I had several problems with detecting possible issues and risks. Because of this I decided to work with prototypes in the second phase. In outsourced projects with their own packages for the complex iterations I tried to find the best way to implement the goal. The prototypes were all separated from the main program and were based on the core functions. Detected risks were, for example, stack overflows during the recursive parsing of big molecules. I had to make the data of each method call smaller and smaller and had to outsource the different actions into several consecutive recursive methods.

4.1.4 Development and Testing

First of all, I tried to implement the prototype one-to-one. Afterwards I tested the new component for all its respective use cases. If one test failed and if I had to change a detail all other tests had to be carried out again. As test tool, I used JUnit[7], a test framework for Java applications.

4.1.5 Planning the Next Iteration

The planning process of a following iteration was the most time-consuming part after the implementation and testing phase. Each step was designed from scratch on paper first. Work flows and procedures of operations were all sketched before I continued to work on. Finally advantages and disadvantages of the different possible capabilities were assessed.

[6] (Oracle, Oracle 2015)
[7] (Wiki 2017)

4.2 Eclipse (IDE)

Eclipse is an open source IDE[8]. The computer programming environment, which is mostly written in Java, supports several programming languages like C, C++, D, Perl, PHP, and Python. Its primary area of operation is the development of Java applications. I have chosen this environment because of its user-friendly operating mode and of its support of various Java-add-ons I needed. I used it for the complete logical in Java written part of the project.

4.3 Dropbox vs ownCloud

The file-hosting service Dropbox[9] founded in 2007 eased my work a lot. I used it to share my data among my other devices and to make it accessible away from my working station. At the beginning of my project I worked with an alternative called ownCloud[10]. Like Dropbox ownCloud is a file-hosting service but in contrast to Dropbox it can be provided by a private server due to its open-source property. Unfortunately, I had to switch platform because of problems with the port forwarding to make it accessible outside the local network.

4.4 JavaFX Scene Builder

The scene builder which JavaFX provides is a graphical tool to design a UI[11] by dragging and dropping the single components. It is possible to modify properties, set style sheets, set Ids and define listeners for event handling. As a result, the tool generates a FXML file (explanation: 5.2.3) with all properties.

4.4.1 Advantages

The design process is simplified a lot by a preview function which shows the window after development without starting the Eclipse-VM[12]. FXML works a lot with margins and paddings saved in CSS. Due to the auto-generated FXML code their structure and syntax is correct.

4.4.2 Disadvantages

During the development process, I had some problems with bugs caused by editing the FXML-file which was not refreshed in the scene builder and afterwards could not

[8] IDE – Integrated Development Environment: tool for software development
[9] (Dropbox kein Datum)
[10] (ownCloud 2017)
[11] UI – User Interface
[12] Eclipse–VM - Eclipse Virtual Machine

be opened as a result of the wrong syntax. Another problem is the runtime visualisation of the components. Depending on the resolution of the screen the scaling was not correct. On the one hand on a 4k display the UI was too small to work with and on the other hand the preview was non-proportional and I had to develop it by parsing the FXML file with the Eclipse-VM.

4.5 Notepad++

Notepad++ is a free text and source code editor. I used it for the development of the CSS layout files. In contrast to Eclipse it has no fixed word lists for autocomplete proposal but it saves foregoing words and code segments in the autocomplete proposal for the current file. This is very time-saving in case Eclipse does not know proposals for FX-components.

4.6 Paint.NET

All illustrations, logos and icons shown in my application were designed with paint.NET. I decided to use the freeware graphic editor instead of Photoshop because of its simplicity in usage and its adaptation to my purposes.

5. Used Technologies

5.1 Java

Java is an object-oriented and class-based programming language released in 1995 by Sun Microsystems. Its syntax is similar to the C and C++ syntax but in contrast to them Java has only a few low-level facilities. Its code can be run on every device supporting the JVM[13] which, beside desktop applications, makes it very useful for client-server and web applications. Due to the JVM it provides WORA[14]. Nowadays it is one of the most popular programming languages with a report of 9 million client-server-developers in 2016 (Wikipedia).

5.1.1 Why Java?

I decided to write my project code in Java due to several aspects. It was important to me that the application is platform-agnostic. The target groups are pupils who work at school or at home. At school users often do not have the permission to install new software and the OS can vary (Linux, Windows, MacOS etc.). A Java application exported to a runnable JAR file has no requirement of installation and can be executed by starting it like all other programs. This makes it easy to copy the JAR application on a USB-stick or to put it in the user’s home directory on the workstation. The object-orientation of Java is another important factor. I wanted to create everything close to reality with objects having a state and properties which they can handle themselves. The logic and event handling is outsourced as much as possible to keep the UI small and understandable.

5.2 JavaFX

JavaFX is a cross-platform framework for the development of Java applications. It was developed by Sun Microsystems (now Oracle Corporation) and released in 2008. With this framework the separation of interactive, multimedia and graphic components is semi pre-implemented due to the MVVM-pattern. Since 2014 JavaFX has intended to replace the GUI[15] toolkits Swing and AWT[16] until then most used.

[13] JVM - Java Virtual Machine
[14] WORA stands for “Write Once, Run Anywhere”
[15] GUI – Graphical User Interface
[16] AWT – Abstract Window Toolkit: GUI widget toolkit like Swing

5.2.1 MVVM

The in JavaFX implemented architectural pattern[17] is a variation of the MVC-pattern. It separates the layout and appearance from the back-end[18] logic. It consists of three components, namely model, view and view model. The model is the content and data-handling component. It manages data modifications and validates new data. It can be developed completely detached from the rest of the application and is reusable. The view-component represents the UI. Compared to the MVC-pattern there is a databinding among view and view model for all manipulations, event handlings and changes in the interface. The view model serves as a connector containing the complete UI-logic but does not know a single information of the view. JavaFX outsources the description of the interface to FXML files, which represent the view. The view model is a controller class with all event-handling-listeners defined in the first element of the FXML file. The main class is now empty except for the initialisation of the FXMLloader, which initialises the controller itself.

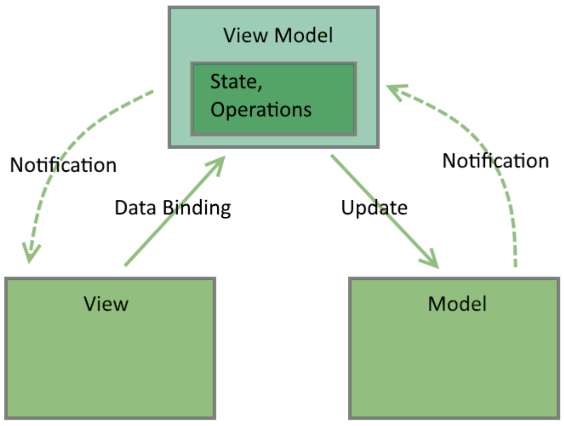


Figure 21: MVVM-pattern

5.2.2 JavaFX vs Swing

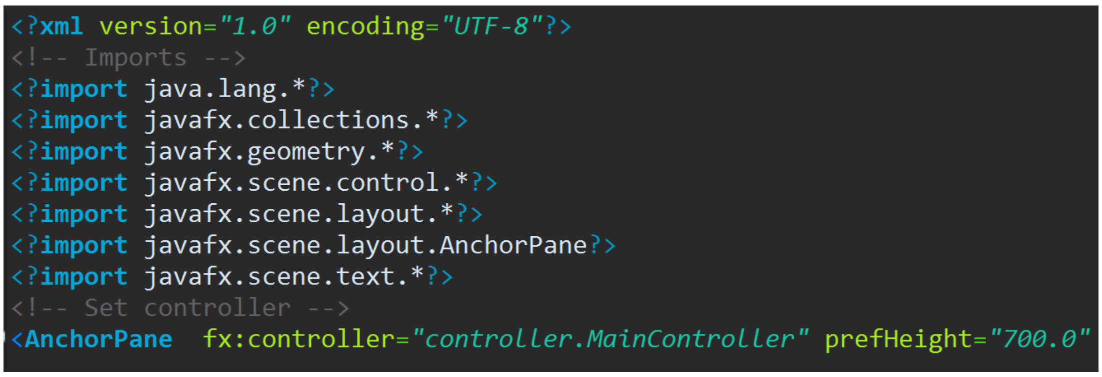
Originally, I decided to write the application with the Swing API. However, there was a huge problem regarding the drawing process. In Swing, each object has a quadratic hitbox[19] and those objects are not rotatable. This is the reason why it is almost impossible to handle precise mouse events. As a first attempt, I tried to write mathematical functions to calculate the real hitbox. I failed at the point where many items did not

[17] Architectural patterns are design patterns for the top-level organisation of interactions and components in a software solution
[18] Back-end – usually cares about business logic and data storage
[19] Hitbox – shape around objects for collision detection

intersect each other not directly but only their oversized hitboxes did. I found the alternative JavaFX, which works with a different kind of paint method and fits the hitboxes directly to the shapes of the objects.

5.2.3 FXML

JavaFX provides two different types of UI-development. One option is to add all knots in their superior knots as it is done in Swing. The other option is to describe the interface in a FXML file. FXML is an XML-based user interface markup language created by Oracle Corporation[20]. To create a valid XML file, it is important that the used components are imported at the beginning of the file.



```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- Imports -->
<?import java.lang.*?>
<?import javafx.collections.*?>
<?import javafx.geometry.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.text.*?>
<!-- Set controller -->
<AnchorPane  fx:controller="controller.MainController" prefHeight="700.0"
```

Figure 22: Imports and controller definiton in FXML

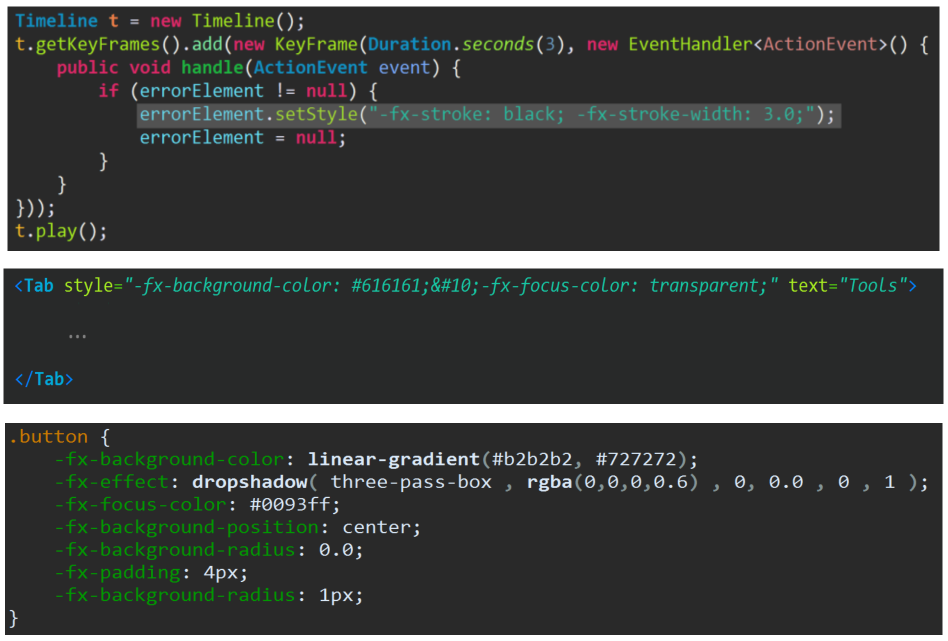
A root pane, usually an anchor pane which can be easily replaced by any other type of pane, wraps the rest of the interface. In the root pane, the controller location is defined. This controller implements the interface Initializable[21], which creates an instance of it after loading the FXML file in the main class.

5.2.4 CSS

Cascading Style Sheet or the short name CSS is a frequently used style sheet language for markup language documents like HTML, XHTML, XML and SVG. The JavaFX CSS follows the W3C standards of CSS with the exception of the prefix of "-fx-"[22]. There are three ways to set a look and feel for an object. FXML elements have the style attribute, JavaFX nodes[23] (javafx.scene.Node) have a "setStyle" method and to outsource the style it can be written in an external CSS file whose file path is set in the main class. I realised that to make the code shorter and understandable it is

[20] (W. Unknown 2017)
[21] (O. Unknown, Oracle 2015)
[22] (Unknown, Oracle 2008, 2015)
[23] (Unknown, Oracle 2008, 2015)

better to use a file to outsource CSS code, for objects that appear often and always have the same style requirement. Therefore, I created an “application.css” file to overwrite the default look of buttons, labels and menus. An important characteristic, which took me a lot of time to understand and to learn how to handle, is that sub-items do not communicate their style to their children or sub items. To handle the priorities of the different setting options Oracle appointed three rules. *A style from a user agent style sheet has lower priority than a value set from code, which has lower priority than a Scene or Parent style sheet. Inline styles have highest precedence. Style sheets from a Parent instance are considered to be more specific than those styles from Scene style sheets.*[24] To understand the functions and capabilities of JavaFX CSS tutorials helped me a lot.[25]



```
Timeline t = new Timeline();
t.getKeyFrames().add(new KeyFrame(Duration.seconds(3), new EventHandler<ActionEvent>() {
    public void handle(ActionEvent event) {
        if (errorElement != null) {
            errorElement.setStyle("-fx-stroke: black; -fx-stroke-width: 3.0;");
            errorElement = null;
        }
    }
}));
t.play();
```

```
<Tab style="-fx-background-color: #616161;&#10;-fx-focus-color: transparent;" text="Tools">

     ...

</Tab>
```

```
.button {
    -fx-background-color: linear-gradient(#b2b2b2, #727272);
    -fx-effect: dropshadow( three-pass-box , rgba(0,0,0,0.6) , 0, 0.0 , 0 , 1 );
    -fx-focus-color: #0093ff;
    -fx-background-position: center;
    -fx-background-radius: 0.0;
    -fx-padding: 4px;
    -fx-background-radius: 1px;
}
```

Figure 23: Differnt ways to set CSS styles

5.3 XML

Extensible Markup Language is a markup language which serves for structured descriptions of hierarchic organised data in form of a text file.[26] I used the XML standard to save the drawings. This guarantees that it is a valid file format and due to this readable with any kind of XML parser. Alternatives to XML would be JSON or setting up a relational database with tables for each kind of element. For this type of data, the effort would be too big because the data model in ChemStruct is pretty simple. If

[24] (Unknown, Oracle 2008, 2015)
[25] (Ebbers 2016)
[26] (Wild, XML: Einführung in XML kein Datum)

later on the project is extended I will probably use Hibernate, a framework for object-related requests.

5.4 JUnit

For testing the functionalities of the single components, I used the testing framework JUnit[27]. Thanks to this tool I fixed several bugs such as the right way to set the coordinates of objects. JUnit guarantees that all functionalities can stand alone by executing the tests in a random order if there are no exceptional annotations.

[27] (Unknown, Wikipedia 2017)

6. Conclusion

6.1 Restrictions

Due to a lack of time I had to cut some features. For the time being the function to change the element type of atoms has been denied. The user can only draw carbon bounds. Another cutback is the limitation of editable files to one if the user does not open the program more than one time.

6.2 Possible expendabilities

In the near future, I would like to implement the features: copy and paste function for objects, multi-window-handling or generally a possibility to open more than one file, a raster function for the drawing pane and the support of multiple languages.

7. Appendix

https://en.wikipedia.org/wiki/Structural_formula

https://docs.oracle.com/javase/8/javafx/api/javafx/scene/shape/Circle.html

https://docs.oracle.com/javase/8/javafx/api/javafx/scene/shape/Line.html

https://docs.oracle.com/javase/7/docs/api/java/lang/Exception.html

https://en.wikipedia.org/wiki/Java_hashCode()

https://en.wikipedia.org/wiki/Toolbar

https://material.io/guidelines/style/color.html#color-color-palette

http://code.makery.ch/library/javafx-8-tutorial/part5/

http://code.makery.ch/library/javafx-2-tutorial/part5/

https://material.io/guidelines/style/color.html#color-color-tool

https://en.wikipedia.org/wiki/Spiral_model

https://de.wikipedia.org/wiki/Spiralmodell

http://istqbexamcertification.com/what-is-spiral-model-advantages-disadvantages-and-when-to-use-it/

http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/is-management/Systementwicklung/Vorgehensmodell/Spiralmodell

https://de.wikipedia.org/wiki/Eclipse_(IDE)

https://en.wikipedia.org/wiki/Eclipse_(software)

https://de.wikipedia.org/wiki/Dropbox

https://owncloud.org/

http://www.oracle.com/technetwork/java/javase/downloads/sb2download-2177776.html

http://www.oracle.com/technetwork/java/javase/downloads/javafxscenebuilder-info-2157684.html

https://en.wikipedia.org/wiki/Notepad%2B%2B

https://de.wikipedia.org/wiki/JUnit

https://en.wikipedia.org/wiki/JUnit

https://en.wikipedia.org/wiki/Paint.NET

https://en.wikipedia.org/wiki/Java_(programming_language)

https://en.wikipedia.org/wiki/JavaFX

https://de.wikipedia.org/wiki/JavaFX

https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel

https://de.wikipedia.org/wiki/Architekturmuster

https://en.wikipedia.org/wiki/Architectural_pattern

https://en.wikipedia.org/wiki/Front_and_back_ends

https://de.wikipedia.org/wiki/Model_View_ViewModel

https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel

https://en.wikipedia.org/wiki/Hitbox

https://en.wikipedia.org/wiki/FXML

http://docs.oracle.com/javafx/2/get_started/fxml_tutorial.htm

https://de.wikipedia.org/wiki/Cascading_Style_Sheets

https://en.wikipedia.org/wiki/Cascading_Style_Sheets

https://docs.oracle.com/javase/8/javafx/api/javafx/scene/doc-files/cssref.html

https://docs.oracle.com/javase/8/javafx/api/javafx/scene/Node.html

https://en.wikipedia.org/wiki/XML

http://code.makery.ch/library/javafx-8-tutorial/part5/

http://code.makery.ch/library/javafx-2-tutorial/part5/

http://www.tutego.de/blog/javainsel/2013/10/deklarative-und-programmierte-ober-flchen-swing-und-javafx-im-vergleich/

https://docs.oracle.com/javase/8/javafx/api/javafx/print/PrinterJob.html

https://en.wikipedia.org/wiki/Skeletal_formula

7.1 Bibliography

Dropbox. *Dropbox.* kein Datum. https://www.dropbox.com/de/?landing=cntl (Zugriff am 26. April 2017).

Ebbers, Hendrik. *GuiGarage.* 09. 02 2016. http://www.guigarage.com/2016/02/javafx-and-css/ (Zugriff am 16. April 2017).

Hill, Graham. *Chemistry Counts.* HODDER AND STOUGHTON, 1987.

Jakob, Marco. *code.makery.* 25. March 2015. http://code.makery.ch/library/javafx-8-tutorial/part5/ (Zugriff am 16. April 2017).

—. *code.makery.* 08. February 2013. http://code.makery.ch/library/javafx-2-tutorial/part5/ (Zugriff am 16. April 2017).

Oracle, Unknown. *Oracle.* 10. 02 2015. https://docs.oracle.com/javase/8/javafx/api/javafx/print/PrinterJob.html (Zugriff am 17. April 2017).

—. *Oracle.* 10. 02 2015. https://docs.oracle.com/javase/8/javafx/api/javafx/animation/Timeline.html (Zugriff am 18. April 2017).

ownCloud. *ownCloud.* 2017. https://owncloud.org/ (Zugriff am 26. April 2017).

Unknown. *Oracle.* 2008, 2015. https://docs.oracle.com/javase/8/javafx/api/javafx/scene/Node.html#setOnZoomFinished-javafx.event.EventHandler- (Zugriff am 16. April 2017).

—. *Oracle.* 2008, 2015. http://docs.oracle.com/javase/8/javafx/api/javafx/scene/doc-files/cssref.html (Zugriff am 16. April 2017).

—. *Wikipedia.* 17. January 2017. https://en.wikipedia.org/wiki/JUnit (Zugriff am 16. April 2017).

Unknown, Oracle. *Oracle.* 11. 01 2016. https://docs.oracle.com/javase/7/docs/api/java/util/prefs/Preferences.html (Zugriff am 17. April 2017).

—. *Oracle.* 10. 02 2015. https://docs.oracle.com/javase/7/docs/api/java/util/prefs/Preferences.html (Zugriff am 18. April 2017).

Unknown, Wikipedia. *Wikipedia.* 14 April 2017. https://en.wikipedia.org/wiki/FXML (accessed April 16, 2017).

Wiki. *Wikipedia.* 17. January 2017. https://en.wikipedia.org/wiki/JUnit (Zugriff am 26. April 2017).

Wild, Michael. „eXtensible Markup Language (XML).“ *Rohdaten.* kein Datum.

—. „XML: Einführung in XML.“ *Moodle.* kein Datum. http://moodle.tfobz.net/pluginfile.php/624/mod_resource/content/2/K01EinfuehrungInXML.pdf (Zugriff am 16. April 2017).