UNIVERSITY OF OSLO
Department of Informatics

Graphical editor for UML 2.0 Sequence Diagrams

Master thesis

Andreas Limyr

27th April 2005
Abstract

The basis for this thesis is the making of a graphical UML 2.0 sequence diagram editor. This editor has been created to experiment with editing behaviors and to get ideas about how to solve different editing conflicts.

The editor was also used to look at the possibility of integration with other tools. The editor was made to generate a file in an existing standard and tested against a tool using this standard.

The thesis shows that it is possible, with limited resources, to develop a sequence diagram editor supporting the new UML 2.0 specification. The thesis will also show a systematic approach to defining editing behavior in a sequence diagram editor.
Acknowledgment

This thesis is submitted to the Department of Informatics at the University of Oslo as part of the Master degree.

I wish to thank my thesis adviser Dr. Øystein Haugen. He has shown an enthusiasm beyond the call of duty. He has given me inspiration and guided me in the right direction time after time.

I would also like to thank Frank Alexander Kraemer, Kai Fredriksen and Jean-Philippe Thibault for their feedback on the SeDi editor.

Finally I would like to thank my family and friends. You have all given me the encouragement and moral support I have needed to finish this thesis.
# Contents

1 **Introduction** .............................................................. 1
   1.1 The domain of interest .............................................. 1
   1.2 Goal of the editor .................................................... 2
   1.3 Goal of the thesis .................................................... 2
   1.4 Thesis structure ....................................................... 2

2 **Background** ................................................................ 4
   2.1 UML 2.0 ................................................................... 4
      2.1.1 Sequence diagrams ............................................... 5
      2.1.2 Changed names .................................................... 10
   2.2 Graphical editors ....................................................... 10
   2.3 Eclipse .................................................................... 11

3 **SeDi editor development** ......................................... 15
   3.1 Syntactical editor ....................................................... 15
      3.1.1 SeDi internal structure ........................................... 16
   3.2 Generation of UML2 models ......................................... 20
   3.3 Status of SeDi ............................................................ 25

4 **Editor integration and use** ..................................... 26
   4.1 Editor integration ....................................................... 26
   4.2 Feedback from users ................................................... 28

5 **Editing behaviors** ..................................................... 30
   5.1 General operations ..................................................... 31
      5.1.1 Horizontal expansion ............................................. 31
      5.1.2 Vertical expansion .................................................. 33
      5.1.3 Undo ................................................................. 36
   5.2 Acceptable/unacceptable Combined Fragment relations .... 36
      5.2.1 Messages ............................................................. 36
      5.2.2 Lifelines ............................................................. 40
5.2.3 Combined Fragments ........................................ 41
5.2.4 Interaction Occurrences .................................. 43
5.3 Combined Fragment editing behavior ..................... 46
  5.3.1 Vertical move of Combined Fragment ................. 46
  5.3.2 Horizontal move of Combined Fragment .............. 54
  5.3.3 Vertical scale of Combined Fragment ................. 63
  5.3.4 Horizontal scale of Combined Fragment .............. 72
  5.3.5 Summary of Combined Fragment editing behavior ...... 79
5.4 Acceptable/unacceptable Interaction Occurrence relations ... 83
  5.4.1 Messages .............................................. 83
  5.4.2 Lifelines ............................................. 86
  5.4.3 Combined Fragments .................................. 87
  5.4.4 Interaction Occurrences ................................ 88
5.5 Interaction Occurrence editing behavior .................. 89
  5.5.1 Vertical move of Interaction Occurrence ............. 89
  5.5.2 Horizontal move of Interaction Occurrence .......... 96
  5.5.3 Vertical scale of Interaction Occurrence .......... 103
  5.5.4 Horizontal scale of Interaction Occurrence ...... 108
  5.5.5 Summary of Interaction Occurrence editing behavior . 112
5.6 Acceptable/unacceptable Lifeline relations ............... 115
  5.6.1 Messages .............................................. 115
  5.6.2 Lifelines ............................................. 116
  5.6.3 Combined Fragments .................................. 118
  5.6.4 Interaction Occurrences ................................ 118
5.7 Lifeline editing behavior .................................. 119
  5.7.1 Vertical move of Lifeline ............................. 119
  5.7.2 Horizontal move of Lifeline .......................... 119
  5.7.3 Vertical scale of Lifeline ........................... 121
  5.7.4 Horizontal scale of Lifeline ........................ 122
  5.7.5 Summary of Lifeline editing behavior ............... 123
5.8 Acceptable/unacceptable Message relations .................. 123
  5.8.1 Messages .............................................. 123
  5.8.2 Lifelines ............................................. 125
  5.8.3 Combined Fragments .................................. 127
  5.8.4 Interaction Occurrences ................................ 127
5.9 Message editing behavior .................................. 127
  5.9.1 Message End move .................................... 129
  5.9.2 Vertical move of Message ............................. 130
  5.9.3 Summary of Message editing behavior ................ 131
6 Conclusions and further work  
   6.1 Conclusions .......................................................... 132  
      6.1.1 SeDi .............................................................. 132  
      6.1.2 Sequence diagram editing analysis ......................... 132  
   6.2 Further work ......................................................... 133  
      6.2.1 SeDi .............................................................. 133  

Bibliography  

A SeDi  
   A.1 SeDi UML2 generation code ........................................ 136  
   A.2 SeDi user manual ...................................................... 146
Chapter 1

Introduction

When the work with this thesis started (spring 2004) the UML 2.0 specification was rather new (fall of 2003). The idea was to take a look at the incorporation of the sequence diagram part of this new standard in development tools. At that point there were no significant tools supporting the new standard. We decided therefore to create our own tool supporting the new standard to perform experiments on. This tool was called SeDi (Sequence Diagram) and was a sequence diagram editor. After creating the tool and performing some experiments there were released new tools supporting the UML 2.0 standard. The SeDi editor was therefore extended with the possibility to export diagrams into a standardized format. These diagrams could then be imported into one of the new tools.

1.1 The domain of interest

This thesis will take a look at the new UML 2.0 specification, with focus on sequence diagrams. We will deal with development of a sequence diagram editor for the new specification. Sequence diagram editors have more functionality than simple box and arrow diagram editors. This is because we are dealing with a language. A language with syntactical and semantical constraints and rules. To have a fully functional editor for a language it should support these constraints and rules. To add an extra level of complexity these constraints and rules for sequence diagram are in two dimensions. That means the placement of elements have significance both in the vertical and the horizontal direction.
1.2 Goal of the editor

SeDi was created to show that it is possible to create a tool for the new UML 2.0 sequence diagram specification. The idea was to make the tool as complete with regards to the specification as possible. In addition to just being an example of a sequence diagram editor SeDi was also made with the intent of being used for experiments. There would be experiments on diagram interchange and tool integration. SeDi was also used as a basis for the discussion about editing behavior. The tool was used as background to find the acceptable and unacceptable relations between the different sequence diagram elements.

1.3 Goal of the thesis

The goal of the thesis is to explore development of tools for sequence diagrams. The idea was to use an open source platform to see if such a tool could be created within existing open source projects.

To create a tool with support for UML 2.0 it is important to define editing behavior. A goal is therefore to find a systematic way to define editing behavior. This means both defining and handling errors.

1.4 Thesis structure

This is a quick overview of the structure of the thesis.

**Chapter 2 Background** This chapter will introduce background for the thesis. First it will look at the UML 2.0 sequence diagram specification, then we will define some different tools for graphic modeling and lastly we will look briefly at Eclipse as a platform for developing a UML 2.0 sequence diagram editor.

**Chapter 3 SeDi editor development** This chapter will go into the details about development of the UML 2.0 sequence diagram editor.

**Chapter 4 Editor integration and use** This chapter will show an example of integration of the SeDi editor with other tools. In addition it will present some of the feedback from users of the tool.

**Chapter 5 Editing behaviors** This chapter will discuss acceptable and unacceptable relations between UML 2.0 sequence diagram elements, how different editing operations can result in unacceptable relations
and how these conflicts can be solved. This part explores what the SeDi editor not yet has implemented.

**Chapter 6 Conclusions and further work** This chapter will summarize chapters 3, 4 and 5. And see how the work done in this thesis can be further explored.

The thesis is rather long in terms of number of pages, but this is mainly because of the abundant amount of pictures. Hopefully the pictures will be helpful in understanding the text.
Chapter 2

Background

2.1 UML 2.0

Making large and complex software systems is not an easy task. The number of lines of code can easily become enormous. Therefore it is a big help to model the system before implementation. When modeled the system is more comprehensible. There are many ways of modeling software systems, but one modeling language stands out. The visual language *Unified Modeling Language* (UML) is an industry standard from *Object Management Group* (OMG) for object oriented system development. It has increased in popularity since it first became standardized in 1997. OMG released a new revision called UML 2.0 in late 2003. UML 2.0 diagrams can be divided into two parts. In the Superstructure Specification [15] these are called *Structure* and *Behavior*. The *Structure* part consists of structural diagrams like class diagrams, component diagrams and deployment diagrams. The *Behavior* part consists of the dynamic diagrams describing behavior. Examples of these diagrams are sequence diagrams, state machines and activity diagrams.

A sequence diagram is a model of the interchange of messages and signals in a system. It can be viewed as a set of traces. More about traces in sequence diagrams can be seen in [8]. It gives the overview of interactions between objects. Although very popular among users the tools for sequence diagrams have not been very advanced.

This thesis has been based on the OMG Final Adopted UML 2.0 Superstructure Specification released in August 2003. A Draft Adopted Specification was released in October 2004. There are some changes of the structure and some of the elements have changed names. See section 2.1.2 on page 10 for a list of some of the changed names. The elements have roughly the same meaning as before. See [15] and [16] to get a complete understanding of the
Figure 2.1: A postman’s everyday life

changes. It is also possible to view [18] and [5] for a more in-depth look at the entire UML 2.0 specification.

2.1.1 Sequence diagrams

We will now look at an example of a sequence diagram. This example will show some of the new constructions of UML 2.0 sequence diagrams.

In figure 2.1 we see the sequence diagram everyday. A sequence diagram can also be called an Interaction. This Interaction shows a postman’s everyday life. The figure named Postman is a Lifeline representing a postman. There are also two other Lifelines in the Interaction. These two Lifelines represent a person and a mailbox.

There is a rectangle with the text loop in the upper left corner. This is
Figure 2.2: A postman's weekday
a Combined Fragment with `loop` as its operator. The area inside the `loop` Combined Fragment is called an operand. The operand in a `loop` Combined Fragment will be repeated a number of times. In our case we will say it repeats in infinity (or at least 40-45 years).

Inside the `loop` Combined Fragment is another Combined Fragment. This Combined Fragment has the operator `alt` (short for alternative). The `alt` Combined Fragment is divided in the middle by a dashed horizontal line. This line separates the operands of the Combined Fragment. The `alt` Combined Fragment has therefore two operands. In an `alt` Combined Fragment at most one of the operands will be performed. In our example this means for every time the loop starts the first operand, the second operand or no operand is performed. As an assumption we can say that the first operand is perform on weekdays and the second operand is performed on weekend days.

Inside the first operand of the `alt` Combined Fragment there is a rectangle with the text `ref` in the upper left corner. This rectangle is an Interaction Occurrence. An Interaction Occurrence is a reference to an Interaction. This particular Interaction Occurrence references the Interaction `weekday`. Interaction Occurrences can be thought of as copying the contents of the referred Interaction into where the Interaction Occurrence is. For this to work the set of Lifelines covered by the Interaction Occurrence and the Lifelines used in the referred Interaction must be the same. In the `everyday` Interaction the Interaction Occurrence referring to the `weekday` Interaction covers the Lifelines Postman, Person and Mailbox. The `weekday` Interaction can be seen in
figure 2.2 on page 6.

It is possible to see that the *weekday* Interaction have the exact same Lifelines as the ones covered by the Interaction Occurrence in figure 2.1 on page 5. In the *weekday* Interaction there is also a Combined Fragment of the type *alt*. This Combined Fragment has two operands. The *alt* operator means that at most one of the operands will be performed.

The two operands represents events where *asynchronous Messages* are sent from the postman to a mailbox. The asynchronous Messages can be thought of as delivering letters and packages to a mailbox. A Message has two *Message Ends*, that is a Message has a sending end and a receiving end. Message Ends connected to Lifelines are called *Event Occurrences*. An Event Occurrence is only the point where a Message is connected to a Lifeline. A Message should normally have a sending end and a receiving end consisting of Event Occurrences. That is a Message is sent from a Lifeline and received by a Lifeline.

But under the *alt* Combined Fragment there are Messages ending and starting in an Interaction Occurrence. Messages were supposed to start and end in Lifelines. But when a Message is connected to an Interaction Occurrence this means the Message goes from one Interaction to another. If we look at the *getMail* Interaction the Interaction Occurrence refers to, we see the same Messages that where connected to the Interaction Occurrence. The *getMail* Interaction can be seen in figure 2.3 on the preceding page. The Message Ends connected to Interaction Occurrences and Interactions are called *Gates*. This means a Message can cross over from one Interaction to another with the help of Gates. Gates are also used if Messages crosses Combined Fragments.

No back to the example. In the *getMail* Interaction we see the mailbox Lifeline get a *call Message*. This Message activates the Lifeline and we can see the activation in the form of a thin rectangle named *Execution Occurrence*. Then the mailbox returns the mail in form of a *reply Message*. The reply Message is sent to the frame of the Interaction and shows up again in the *weekday* Interaction.

At the bottom of the Interaction *weekday* we can see yet another Combined Fragment, this time with the *opt* operator. This means the Combined Fragment is optional and the operand inside the Combined Fragment is not always performed. Inside the only operand is an Interaction Occurrence. This refers to the Interaction *complaint*. This Interaction can be seen in figure 2.4 on the next page.

In the *complaint* Interaction the person complains to the postman about being late. The postman then makes different excuses (or non at all). This is good enough for the person and the conversation is over.
Figure 2.4: A person complaining about the postman being late
Table 2.1: Changed names

<table>
<thead>
<tr>
<th>Old name</th>
<th>New name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interaction Occurrence</td>
<td>Interaction Use</td>
</tr>
<tr>
<td>Stop</td>
<td>Destruction Event</td>
</tr>
<tr>
<td>Event Occurrence</td>
<td>Occurrence Specification</td>
</tr>
<tr>
<td>Execution Occurrence</td>
<td>Execution Specification</td>
</tr>
</tbody>
</table>

This ends the complaint Interaction and we can go back to the weekday Interaction. There are no more event under the opt Combined Fragment in the weekday Interaction so this concludes the weekday Interaction too. Then we go back to the initial Interaction everyday seen in figure 2.1 on page 5. There we find ourselves in the first operand of the Combined Fragment. The other operand shows an Interaction Occurrence referring to the Interaction weekend. We will not go into this scenario, but we can see that the postman does not interact with mailboxes in the weekend.

2.1.2 Changed names

For readers of the Draft Adopted UML 2.0 Superstructure Specification the names may be confusing. Table 2.1 shows some of the new names of the most often mentioned elements. The elements have changed somewhat, but the differences are minor. See [15] and [16] to get a complete understanding of the changes.

2.2 Graphical editors

We will try to categorize different graphical editors. We will start with the most basic graphical editor. This is a drawing editor. This is an editor with the ability to draw different lines and shapes. Microsoft Paint and Xfig are examples of simple drawing programs. They are not created with the intention of supporting a certain modeling language, just the ability to draw. It is possible to draw UML 2.0 diagrams using these tools, but there are no restrictions or support for creation of UML 2.0 diagrams in particular. We will call these graphical editors for drawing editors. They know nothing about the syntax or semantics of any modeling language.

The next level is graphical editors with support for a certain modeling language. Support will be in the form of templates for the figures in the modeling language. This means that complex figures are ready to be pasted
into the diagram editor. These figures can also have support for gluing and stretching. An example of this form of editor is Visio [3]. It is possible to draw UML 2.0 diagrams using a template for UML 2.0 in Visio. This template can consist of the different sequence diagram figures. It can have the possibility of gluing different parts together and allowing customized editing operations specially created for the different figures. It can be said to support the graphical syntax of a modeling language. Therefore we will call these graphical editor for syntactical editors. A syntactical editor can therefore allow semantically illegal diagrams.

The third level of graphical editors is editors with support for checking the semantic of the diagram. This means checking if semantic constraints are violated. If a constraint is broken the editor should do one of two things, either undo whatever edit operation led to the violation of the constraint or modify the diagram in such a way that the constraint is not longer broken. We will call these editors for semantical editors. Examples of semantical editors with support for UML 2.0 are IBM Rational Software Architect and Modeler and Telelogic TAU/Generation2. What these two different tools offer can be seen in [12] and [1]. With support for semantics it is also possible to generate meaningful code and documentation. This increases the effectiveness of modeling and can reduce the time from model to application greatly.

2.3 Eclipse

“Eclipse is a kind of universal tool platform - an open extensible IDE\textsuperscript{1} for anything and nothing in particular.”

\begin{flushright}
\textit{www.eclipse.org}
\end{flushright}

Eclipse is an open platform intended to overcome the differences in operating systems and to support creation and execution of tools for software development (or anything else!). The Eclipse platform is written in the Java programing language and can therefore run on most operating system with a \textit{Java Runtime Environment} (JRE). A list of operating systems supported by Eclipse can be found in [4].

The platform can be used for ordinary Java development, but the real strength is the extensibility of the platform. Extending Eclipse is done by creating plug-ins. These plug-ins are then integrated with the platform core.

\textsuperscript{1}Integrated Development Environment
An illustration of this can be seen in figure 2.5 on the facing page. To learn more about plug-in development for Eclipse see [6], [4] and [9].

The sequence diagram editor described in 3 on page 15 was made as a plug-in to Eclipse. This was done because Eclipse offers frameworks for both graphical modeling and creation of UML 2.0 repositories.

A framework is a reusable design expressed as a set of abstract classes and the way their instances collaborate. It is a reusable design for all or part of a software system. By definition, a framework is an object-oriented design. It does not have to be implemented in an object-oriented language, though it usually is. Large-scale reuse of object-oriented libraries requires frameworks. The framework provides a context for the components in the library to be reused.

The Graphical Editing Framework (GEF) allows developers to take an existing application model and quickly create a rich graphical editor. GEF consists of two Eclipse plug-ins, draw2d and gef. The draw2d plug-in provides a layout and rendering toolkit for displaying graphics. The gef plugin employs a MVC (model-view-controller) architecture which enables simple changes to be applied to the model from the view. The developer can then take advantage of the many common operations provided in GEF and extend them for the specific domain.

MVC was first described in 1979 by Trygve Reenskaug [17], then working on Smalltalk at Xerox parc, it is also described in [14]. It is a pattern that divides the data model of the application from the graphical view. The controller is responsible for processing events (like user interaction) and updating the model according to these events. The model can then notify the view of changes in the model. Simplified it can be said to be a triangle with the model, view and controller communicating. The view is just a way of presenting the model. MVC comes in many forms and implementations, but the main structure is something like this.

In GEF MVC is implemented with the following strategy: Bring your own model and present the model using the draw2d plug-in. The gef plug-in provides the means for communicating between the model and the draw2d plug-in (the view). Therefore the gef plug-in can be seen as the controller. An introduction to GEF can be found in [10].

The Eclipse Modeling Framework (EMF) is also a framework. EMF provides a metamodel for the creation of data models. A metamodel is the underlying structure that defines the language for expressing a model. In other words, the UML metamodel is a model that is used to define the UML. More about UML metamodels can be seen in [13]. The UML2 project is an EMF-based implementation of the UML 2.0 metamodel for the Eclipse platform. UML2 is able to create UML 2.0 data models based on
Figure 2.5: Eclipse Platform architecture
the metamodel and has support for creating an XMI schema from the data models. UML2 gives therefore the possibility to create UML 2.0 data models in a standardized way. And this allows interchange of UML 2.0 data models. To learn more about UML2 view [11].

Using both GEF and UML2 gives the framework to create a graphical editor with the possibility to create standardized UML 2.0 models.
Chapter 3

SeDi editor development

A graphical sequence diagram editor was made to perform experiments. This editor was made using the Eclipse platform, GEF and UML2 mentioned in 2.3 on page 11. The goal of making the editor was the ability to experiment with diagram interchange and tool integration and to look at different editing behaviors of elements in a sequence diagram. This part will shortly explain the development process of the SeDi (Sequence Diagram) editor.

3.1 Syntactical editor

The first iterations of the editor had emphasis on creating a tool with the equivalence of a syntactical editor (see section 2.2 on page 10 for the definition).

It was decided early on to implement a subset of the elements found in the UML 2.0 sequence diagram specification. This was because of the time limit for completing the thesis. The elements left out of the editor are; General Ordering, Interaction Constraints, Part Decomposition and State Invariant. The elements were not implemented because they really do not add much to the exploration of this thesis.

General Ordering is a special kind of Message, Interaction Constraints is a constraint in form of a text covering a Lifeline, Part Decomposition is a reference inside the head of a Lifeline and State Invariant can be presented in the same way as an Interaction Constraints.

These elements are not challenging graphically so it would be an easy task incorporating these elements into the SeDi editor, but the time limit forced these elements to be left unimplemented.

The editor supports the nine graphical elements mentioned in table 3.1 on the following page. An example with all these elements can be seen in fig-
Table 3.1: UML 2.0 sequence diagram elements supported by the SeDi editor.

<table>
<thead>
<tr>
<th>Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interaction</td>
</tr>
<tr>
<td>Lifeline</td>
</tr>
<tr>
<td>Message (three types)</td>
</tr>
<tr>
<td>Combined Fragment</td>
</tr>
<tr>
<td>Interaction Occurrence</td>
</tr>
<tr>
<td>Execution Occurrence</td>
</tr>
<tr>
<td>Continuation</td>
</tr>
<tr>
<td>Stop</td>
</tr>
<tr>
<td>Coregion</td>
</tr>
</tbody>
</table>

Therefore creating a syntactical editor for UML 2.0 sequence diagrams was rather straightforward. The different elements do not have too complex graphical representations. In addition to that the editing operations needed were rather basic.

### 3.1.1 SeDi internal structure

As mentioned in section 2.3 on page 11 the framework the SeDi editor is built on, GEF, is based on a MVC principle. In the SeDi editor this is implemented with splitting each element into a model part and a view part. GEF then supports the means for communication between these parts with a controller. A simplified example of this communication can be seen in figure 3.2 on page 18. The diagram shows how moving of a Lifeline is performed inside...
Figure 3.1: An example diagram with all the elements supported by the SeDi editor
SeDi. A controller processes the user event *move Lifeline*. The LifelineModel gets a new location, then the LifelineModel notifies the controller that it has changed location. The controller then gets the new location from the LifelineModel and finally the LifelineView is drawn with the new location.

This shows that each element has both a model part and a view part. But how are the different model elements connected? That is, how is for example a Lifeline model part associated with a Combined Fragment model part. The structure of the SeDi editor model is quite simple. All model parts, except Messages, are children of the Interaction. The sequence of creation of the different elements is what decides the layering of the different elements. If an Execution Occurrence is created before a Lifeline, the Execution Occurrence will be covered graphically by the Lifeline. This means the only relation the model elements share is the same parent Interaction.

When it comes to Messages they are a bit different than the other elements. A Message does in fact have direct relations with the two elements it is connected to.

In figure 3.3 on the facing page there is an example of how the same SeDi diagram is represented graphically and internally.
(a) A SeDi diagram

(b) A data-structure representation of the diagram in (a)

Figure 3.3: Example of the data-structure in the SeDi editor
3.2 Generation of UML2 models

With the syntactical editor version of SeDi it would be possible to draw UML 2.0 sequence diagrams. At this stage the tool could not support any form of interchange of the diagrams with any other tool than itself. This limits the usefulness considerably. By creating diagrams other applications can interpret, it is possible to benefit from these applications as well. The Eclipse UML2 project supports a standard for creating UML 2.0 models in an XMI schema. See section 2.3 on page 11 and [11] for more details about the UML2 project.

It was necessary to parse the model created in SeDi in such a way that a model based on UML2 could be created. We will call the editor-created model for the graphical model and the model based on UML2 for the UML2 model. As described in 3.1.1 on page 16 relations between the different model elements in the graphical model are limited to relations between the Interaction and its children. The only exception is that Messages are associated with the two elements they are graphically connected to.

Creation of the UML2 model will therefore mean making the necessary associations not yet present in the graphical model and creating certain elements not present in the graphical model. We will take a closer look at how this was done. We will present an example of how a diagram made in the SeDi editor is parsed to create a UML2 model. The example we will use can be seen in figure 3.4 on the next page. The graphical model of this diagram is presented in figure 3.5 on the facing page.

To create a UML2 model of this diagram it is necessary to first create a Model. The Model gets the same name as the graphical model file. Let us say that our example diagram file is somename.sd, then the UML2 model is created like this:

```java
Model model = UML2Factory.eINSTANCE.createModel();
model.setName("somename");
```

We would like to create an Interaction in the UML2 model to represent our diagram. But to do so we must first create a Collaboration. This is because an Interaction fits into a bigger picture in the UML2 model than in the graphical model. See [7] for more about how Interactions are connected to the rest of the UML.

With a Collaboration we can create the Interaction. We will use the same name as the graphical Interaction, "example".

With a UML2 Interaction we can get started with creation of UML2 Lifelines. To do this we need to find all graphical Lifelines. The graphical Interaction has two Lifelines so we need to create two UML2 Lifelines.
Figure 3.4: A diagram made with the SeDi editor

Figure 3.5: The graphical model of the diagram in figure 3.4
But to create a UML2 Lifeline the UML2 model demands the Lifeline represents a Class. So based on the graphical Lifeline it must first be created a UML2 Class. But a UML2 Lifeline does not directly represent the Class. We must also create a Property. This Property is a link from a Class to an instance of that Class. That is a Property is an instance of a Class. And a Lifeline represents this instance. The creation of both Class and Property is based on the text in the head of the graphical Lifeline.

This is done for the two Lifelines in our example. With two Classes and two Properties we can now create the UML2 Lifelines.

In our example the graphical Lifeline c1:C will therefore generate a Class, a Property and a UML2 Lifeline. This can be seen in figure 3.6. Here we see a diagram only consisting of Lifeline c1:C and the UML2 model created from that diagram.

So now we have UML2 Lifelines. The next step is now to create the remaining elements. To do this we must create some sort of order among the remaining elements from figure 3.5 on the preceding page. The remaining elements are a Combined Fragment, an Interaction Occurrence and two Messages. First all elements inside a Combined Fragment are connected to...
this Combined Fragment. Then SeDi sorts the remaining elements according to their height value. The highest elements first. If two elements have the same height value, the leftmost element sorted first. This is the way SeDi organizes its elements.

In our example this means the Interaction Occurrence and the Message inside the Combined Fragment are connected to the Combined Fragment. Then the elements not contained inside the Combined Fragment are sorted. In our example we sort the Combined Fragment and the Message. The Combined Fragment is sorted first, then the Message. With this list over events we can start creating the Combined Fragment.

The UML2 Combined Fragment must refer to the Lifelines it covers. This is found out using the graphical Combined Fragment and the graphical Lifelines. Then the operator of the UML2 Combined Fragment is set to `alt`. And then the operand is created and it covers the same Lifelines as the Combined Fragment. This means the graphical Combined Fragment is turned into one UML2 Combined Fragment and one UML2 operand. If the graphical Combined Fragment was separated by separators there would be more operands.

After creating the operand the elements contained in the operand (Interaction Occurrence and Message) are created. Before they are created they are sorted according to the left top point. This means the Interaction Occurrence is the next element to be created.

The UML2 Interaction Occurrence is created and set to cover the Lifelines it covers. Then the UML2 Interaction Occurrence name is set. The name is the same as that of the graphical Interaction Occurrence. This means the graphical Interaction Occurrence only generates a UML2 Interaction Occurrence.

The next element to be created is the Message inside the Combined Fragment. The graphical Message is already connected to the Combined Fragment and the Interaction Occurrence (see figure 3.5). A UML2 Message is created. Then the Message name and sort is set. In addition to that the UML2 Message needs Message Ends. In this example there are created two Gates, one Gate connected to the Combined Fragment and one connected to the Interaction Occurrence. This means the graphical Message turns into one UML2 Message and two Message Ends. In the example these Message Ends are Gates.

And that was the elements inside the Combined Fragment. This means we are back to the list of elements outside the Combined Fragment. We have created the Combined Fragment and the next element is the Message. The UML2 Message is created, the name and sort set. Then the two Message Ends must be created. In this example there is created an Event Occurrence connected to the Lifeline c1:C and a Gate connected to the Combined
Figure 3.7: The UML2 model of the diagram in figure 3.4
See figure 3.7 on the preceding page to get an overview of the UML2 model created from the example seen in 3.4.

For a closer look at the code it is possible to view appendix A.1 on page 136.

### 3.3 Status of SeDi

As a conclusion it can be said that the SeDi editor is a syntactical editor with the possibility to generate a UML2 repository of its diagrams. This means SeDi support creation of sequence diagrams, but not all constraints are checked. This can lead to creation of diagrams not entirely correct. But the SeDi editor does in fact support a number of constraints, like Messages can only connect to Lifelines, Interaction Occurrences, Combined Fragments and Interaction. And nothing can be created outside of an Interaction.
Chapter 4

Editor integration and use

This part will deal with a test of integrating the SeDi editor with another modeling tool. It will also outline possible extension made to SeDi making it more integrated with another tool.

This part will also present some feedback from users.

4.1 Editor integration

As described in section 3.2 on page 20 the SeDi editor was extended to create a file in the UML2 format. This opens for the possibility to create sequence diagrams in SeDi and export them to other tools supporting this standard. With diagram interchange it is possible to benefit from all the tools using the same format.

The UML2 file that SeDi creates is in a format IBM RSM (Rational Software Modeler) also uses. It is possible to import a UML2 file created in SeDi into RSM. Then RSM creates a project from the information in the UML2 file. Then RSM can visualize the sequence diagram made in SeDi. A diagram made in SeDi and the same diagram imported to RSM can be seen in figure 4.1 on the facing page.

The problem with importing a SeDi created UML2 file into RSM is the fact that it does not automatically integrate with an existing RSM model. Even though the UML2 file can have the same classes (they have the same names) as an existing RSM model, the import will not treat the classes as the same. That is if the existing RSM model has a class Person and the UML2 file has a class Person the import will deal with these two classes as different. This has to do with internal id-values in the UML2 file and the RSM model.

This could be overcome with some alterations. Merging the UML2 file
Figure 4.1: The same sequence diagram in the SeDi editor and RSM
and the RSM model could be done by substituting id-values. In the example with two Person classes it could be possible to just substitute the id-values of one of the classes with the id-values of the other.

The ideal situation would be to use SeDi as the native sequence diagram editor inside RSM. This would mean SeDi would replace the sequence diagram editor that comes with RSM. This would mean SeDi would have to have full access to the RSM model. Every change made in SeDi would instantly be updated in the RSM model.

This would give the possibility to get all the benefits of a big platform like Rational Software Modeler while still using SeDi.

4.2 Feedback from users

The SeDi editor has been distributed to a small number of selected people. These people can roughly be divided into two groups, students and researchers.

SeDi was used in the course INF2120 at the Department of Informatics at the University of Oslo in the spring of 2005. And this lead to some feedback from the students that used SeDi.

The feedback from students concentrated on two things - usability and UML2 integration.

When it came to usability it was a desire that there were more standardized shortcuts. It was also a wish that all buttons for creation of new elements behaved in the same way.

The feedback about UML2 integration was rather clear. If SeDi was totally integrated with the Eclipse UML2 project the usefulness would be greater. This would lead to that the SeDi editor could instantly benefit from tools created for the UML2 project. And this would also lead to the possibility to import UML2 files into the SeDi editor.

The feedback from researchers also concentrated on UML2 integration. In addition to that there were some feedback about the user guide.

The researchers were Frank Alexander Kraemer (NTNU), Jean-Philippe Thibault (IRISA) and Ina Schieferdecker (Fraunhofer FOKUS).

When it came to UML2 integration it was pretty much the same as the students. It was a common request for support for UML2 files as input. That is creating sequence diagrams in SeDi from a UML2 file.

There were also some feedback about the user guide. It was suggested that the user guide should be more extensive. Maybe in the form of a step-by-step tutorial. The user guide can be seen in appendix A.2.
A researcher also found the tool too limited because it only dealt with the sequence diagram part of UML 2.0.
To summarize the feedback it is possible to say that the most wanted addition to the SeDi editor was the ability to take UML2 files as input. This and a closer integration with the Eclipse UML2 project as a whole was what most users wanted.
Chapter 5

Editing behaviors

This part will discuss editing behaviors of some of the constructs found in UML 2.0 sequence diagram.

First of we will define a couple of general operations to use if it is necessary to perform some kind of modification on the diagrams.

To limit the scope of this thesis we will concentrate on Combined Fragments, Interaction Occurrences, Lifelines and Messages. For each of the constructs we will establish acceptable and unacceptable relations with other elements.

Then we will look at different editing operations for the elements. The operations we will look at are move and scale. Both move and scale will be decomposed into vertical and horizontal.

For each editing operation we will discuss the effects on other elements in the sequence diagram. We will work under the assumption that the original diagram is correct. If the edit generates an unacceptable relation with another element it will be discussed how to resolve this conflict. After looking at the different elements there will be drawn a summarizing conclusion about this edit operation for the element discussed.

There are a few assumptions made for the discussions. This is to limit the amount of special cases to handle.

1. All Lifeline heads are aligned

2. There are no Messages to and from the same Lifeline

3. Interaction Occurrences and Combined Fragments are always placed under the heads of the Lifelines.

4. All Gates on Interactions are placed under the heads of the Lifelines.
The first assumption will limit vertical move of Lifelines. It will also limit vertical scale of the top of Lifelines.

The third assumption will inhibit vertical movement and vertical scale of Combined Fragments and Interaction Occurrences. All movement and scaling ending in a Combined Fragment or an Interaction Occurrence no longer being placed under Lifeline heads will be disallowed. In the further discussion we will assume this restriction is already implemented.

The first, third and fourth assumption will also affect the placement of Messages. With these three assumptions all Message Ends must be placed under the heads of Lifelines. This is because all possible placement of Message Ends have been constrained to be under the heads of Lifelines. This will prevent Messages intersecting Lifeline heads.

5.1 General operations

When performing an edit it is often desirable to do some changes to the diagram to resolve conflicts created by the edit. Therefore some general operations for modifying a diagram can be useful. It is important that the operations used does not result in other conflicts. Therefore it would be nice to have some general operations that can be used and not create any more problems.

A common problem is too little room to perform an edit. This suggests an automatic operation for making room. It is possible to divide such an operation into horizontal and vertical expansion.

In addition to that we can sometimes come in a unsolvable situation. Then we will use the undo operation.

5.1.1 Horizontal expansion

Horizontal expansion will mean making room between Lifelines. This will be done by pushing things to the right. Imagine a vertical line in a diagram (see (a) of figure 5.1 on the next page). Place this line between two Lifelines where more room is needed. Then imagine the line splitting into two vertical lines. One of the lines will be in the same place as the original. The other will move to the right. When the line moves it also pushes everything to the right of it. It pushes the elements in such a way that the position of everything on the right side of the line constantly has the same distance to the line. Elements with points on both sides of the initial expansion line will be stretched (see (b) of figure 5.1 on the following page).
It should only be possible to place the expansion line between Lifelines, and not in such a way that it crosses the head of a Lifeline. This operation will only stretch elements connected to more than one Lifeline.

From the subset of elements in the SeDi editor (see section 3.1) there are only four elements that can cover more than one Lifeline. These elements are Messages, Combined Fragments, Continuations and Interaction Occurrences.

Because the expansion is only expanding in a horizontal direction each element affected can be viewed separately. There will be no change in the order of the elements on the lines of the Lifelines. And because we can view each element separately there will be no conflicts created between existing elements. But will there be conflicts in stretching the different elements?

Combined Fragments, Continuations and Interaction Occurrences can all be viewed as rectangular shapes. Rectangles can be stretched in a horizontal way without loosing its shape. Therefore we can perform horizontal expansion without generating problems with Combined Fragments, Continuations and Interaction Occurrences.

Messages on the other hand can create problems because the angles between the Message and the two elements the Message is connected to can...
change. In (b) of figure 5.2 on the next page this happens and the result of the horizontal expansion is that the Message intersects with the Interaction Occurrence. To prevent this we will insert a horizontal line into the Message at the same place as the horizontal expansion is performed. This is done in (c) of 5.2. With the use of this modification of the Message the expansion operation can be performed without generating problems. The expanded area will only contain Messages with horizontal lines. And the area outside of the expanded area is the same as before.

In the further discussion we will refer to this as horizontal expansion.

5.1.2 Vertical expansion

Vertical expansion will make room on the lines of the Lifelines. It will have to push everything under a given horizontal line down-wards. Think of it as inserting the same amount of unused Lifeline line into the same place for all Lifelines. The problem with this solution is elements with points over and under the expansion line.

Elements with points over and under the expansion line must be stretched. It is possible to think of the initial expansion line as first one line (see figure 5.3 on page 35), then it splits into two (see figure 5.4 on page 35). The two lines represents the two sides of the initial line. The line representing the upper side will be in the same position as the original, but the underside will move down-wards. The elements will stretch with the down-wards moving line.

Stop and Continuation will be treated like they only have one point, it will be the center of the figure. In the example seen in figures 5.3 to 5.4 on page 35 Stop and Continuation are both over the expansion line. Therefore they will not be affected. Of the elements mentioned in section 3.1, table 3.1 on page 16, only four can be said to have points over and under the horizontal expansion line. These elements are Messages, Combined Fragments, Interaction Occurrences and Execution Occurrences.

Combined Fragments, Interaction Occurrences and Execution Occurrences can all be viewed as rectangular shapes. Since the rectangles will cover the same horizontal area after a vertical expansion they will not interfere with more elements than before. Therefore Combined Fragments, Interaction Occurrences and Execution Occurrences can safely be stretched.

Messages can create problems. If the Message points down-wards and one Message end is over and the other is under the expansion line there can be conflicts. This is the same problem encountered in horizontal expansion, so we will use the same solution as we did there. Instead of inserting a horizontal line we will insert a vertical line.
Figure 5.2: Creating an intersect situation and a possible solution
Figure 5.3: Before vertical expansion down-wards from the solid line.

Figure 5.4: After vertical expansion down-wards between the solid lines.
In the further discussion we will refer to this as *vertical expansion*.

5.1.3 Undo

The *undo* operation will do just that. It will undo the edit and restore the diagram to the exact same condition as before the edit.

5.2 Acceptable/unacceptable Combined Fragment relations

To help in the discussion about editing behavior of Combined Fragments we look at the possible relations a Combined Fragment can have with other elements. We will decide what is acceptable and what is unacceptable. To decide this we will look in the UML 2.0 specification [15] for constraints. But because the UML 2.0 specification mostly concentrates on the meta-model and not the graphical we will also create our own graphical constraints. This is done to guarantee the readability of the diagram. Ambiguities or difficulty separating elements are examples of reduced readability. We will only create additional constraints that goes beyond the specification, that is we will first see if a relation is unacceptable in the specification before we create an additional constraint.

To differentiate if the relation is unacceptable because of the specification or our additional constraints we will create two unacceptable categories. If the relation is unacceptable according to the specification it will be a *specification breaking* relation. If it is unacceptable because of one of our additional constraints we will refer to it as a *readability reducing* relation.

5.2.1 Messages

There are six relations a Message can have with a Combined Fragment.

1. A Message inside a Combined Fragment and not connected to the Combined Fragment
2. A Message inside a Combined Fragment and connected to the Combined Fragment
3. A Message outside a Combined Fragment and not connected to the Combined Fragment
4. A Message outside a Combined Fragment and connected to the Combined Fragment

5. A Message connected to and intersecting a Combined Fragment

6. A Message not connected to and intersecting a Combined Fragment

The first relation is an entire Message inside a Combined Fragment. See (a) of figure 5.5 on the following page for an example of this. This is acceptable according to the specification. It does not clutter the diagram to have Messages inside a Combined Fragment and it is meaningful, therefore we will not create an additional constraint. The relation is acceptable.

The second relation is a Message inside a Combined Fragment and directly connected to the Combined Fragment (i.e. it has a Gate), but as long as it does not intersect the Combined Fragment this is also allowed according to the specification. See (b) of figure 5.5 on the next page for an example. There are no reasons with regards to the readability of the diagram for disallowing this. This relation is therefore acceptable.

The third relation is a Message outside a Combined Fragment and not connected to the Combined Fragment. See (c) of figure 5.5 on the following page as an example. This is acceptable according to the specification. And it is highly readable so we will let this relation be acceptable.

The fourth relation is a Message outside a Combined Fragment and the Message starts or ends in the Combined Fragment. An example can be viewed in (b) of figure 5.5 on the next page. The specification does not disallow this. There are no reasons with regards to the readability of the diagram for disallowing this. The relation is therefore acceptable.

The fifth relation is a Message connected to and intersecting the Combined Fragment at the same time. See (d) of figure 5.5 on the following page as examples. This is not acceptable according to the specification because a Message cannot cross boundaries of Combined Fragments. This is listed as constraint number seven in section 14.3.14 Messages in [15]. If this happens we have a specification breaking relation between a Combined Fragment and a Message. We will interpret the meaning of boundaries of Combined Fragments to consist of the outer frame of the Combined Fragment along with the separator dividing the Combined Fragments operands.

The sixth relation is a Message not connected to and intersecting the Combined Fragment at the same time. That is parts of the Message are enclosed by the Combined Fragment and other parts are not. See (e) of figure 5.5 on the next page as examples. This is not acceptable according to the specification because a Message cannot cross boundaries of Combined Fragments. This is therefore a specification breaking relation.
Figure 5.5: Combined Fragment in relation with Message
<table>
<thead>
<tr>
<th>Relation</th>
<th>Assessment</th>
<th>Reject reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Message inside a Combined Fragment and not connected to the Combined Fragment</td>
<td>Acceptable</td>
<td>-</td>
</tr>
<tr>
<td>A Message inside a Combined Fragment and connected to the Combined Fragment</td>
<td>Acceptable</td>
<td>-</td>
</tr>
<tr>
<td>A Message outside a Combined Fragment and not connected to the Combined Fragment</td>
<td>Acceptable</td>
<td>-</td>
</tr>
<tr>
<td>A Message outside a Combined Fragment and connected to the Combined Fragment</td>
<td>Acceptable</td>
<td>-</td>
</tr>
<tr>
<td>A Message connected to and intersecting a Combined Fragment</td>
<td>Unacceptable</td>
<td>Specification breaking</td>
</tr>
<tr>
<td>A Message not connected to and intersecting a Combined Fragment</td>
<td>Unacceptable</td>
<td>Specification breaking</td>
</tr>
</tbody>
</table>

Table 5.1: Combined Fragment/Message relations
For a summary of the discussion about acceptable/unacceptable Combined Fragment/Messages relations see table 5.1 on the preceding page. The table shows four acceptable relations and two unacceptable relation.

### 5.2.2 Lifelines

There are three relations a Combined Fragment can have with a Lifeline:

1. A Combined Fragment covering a Lifeline
2. A Combined Fragment not covering a Lifeline
3. A Combined Fragment intersect or enclose the head of the Lifeline.

Because of the assumption that Combined Fragments always must be under Lifeline heads a Combined Fragment will never intersect or enclose the head of a Lifeline. We will therefore disregard the third relation.

A Combined Fragment covers a Lifeline if the line of the Lifeline is intersected by the Combined Fragment. If it is not intersected it is not covered.

The first relation can be seen in figure 5.6. There the Combined Fragment covers Lifeline c2. There are no constraints in the specification making this an unacceptable relation. There is no reason with regards to the readability of the diagram we should disallow this. Therefore we will view this as acceptable.

The second relation is a Combined Fragment not covering a Lifeline. An example can be viewed in figure 5.6. There the Combined Fragment has this relation with Lifeline c1. The specification allows this and there are no apparent reasons for creating new constraints disallowing this. This relation is therefore acceptable.
Table 5.2: Combined Fragment/Lifeline relations

Table 5.2 summarizes the discussion about acceptable/unacceptable relations between Combined Fragments and Lifelines.

### 5.2.3 Combined Fragments

There are four different relations a Combined Fragment can be in with regards to another Combined Fragment.

1. A Combined Fragment inside a Combined Fragment
2. A Combined Fragment contains a Combined Fragment
3. Two Combined Fragments outside of each other
4. Two Combined Fragment intersects each other

The first relation is a Combined Fragment inside another Combined Fragment. A Combined Fragment is inside another when all four corners of the contained Combined Fragment is inside the four corners of one of the containing Combined Fragments operands. An example of this can be seen in (a) of figure 5.7 on the following page. This means that both the `alt` and `loop` Combined Fragments are inside the `par` Combined Fragment. The `loop` Combined Fragment is also inside the `alt` Combined Fragment. The specification allows this. It is both meaningful and very readable to have this relation, therefore we will make this an acceptable relation.

The second relation is one Combined Fragment containing another Combined Fragment. When it comes to the contains relation it has the same definition as the inside relation, only this time the Combined Fragment is the container. There are no constraints in the specification and we will not create new ones. Therefore this relation is acceptable.

The third relation is outside. An example of a Combined Fragment being outside another Combined Fragment is (b) of figure 5.7 on the next page. Here the `loop` Combined Fragment has the outside relation to both the `opt`
Figure 5.7: Combined Fragment in relation with Combined Fragment
Combined Fragment and the *alt* Combined Fragment. Combined Fragments in this relation can not contain each other. They can however be contained in other Combined Fragments. This is acceptable according to the specification and there are no reasons for making it unacceptable from a readability viewpoint. Therefore this relation is acceptable.

The fourth and last relation is Combined Fragments intersecting. This goes for intersecting the border of a Combined Fragment as well as the borders of its operands. Examples of intersection can be seen in (c) and (d) of figure 5.7 on the facing page. This relation is not mentioned in the constraints of the specification. But it makes the diagram harder to read. We will therefore suggest this be a readability reducing relation.

The discussion about acceptable/unacceptable relations between Combined Fragments resulted therefore in three acceptable relations and one unacceptable relation. See table 5.3 for the complete result.

### 5.2.4 Interaction Occurrences

Combined Fragments can have four different relations to Interaction Occurrences:

1. A Combined Fragment inside an Interaction Occurrence
2. A Combined Fragment contains an Interaction Occurrence
3. A Combined Fragment and an Interaction Occurrence outside of each other
4. A Combined Fragment and an Interaction Occurrence Intersects each other
Figure 5.8: Combined Fragment in relation with Interaction Occurrence
A Combined Fragment contains an Interaction Occurrence

A Combined Fragment and an Interaction Occurrence outside of each other

A Combined Fragment inside an Interaction Occurrence

A Combined Fragment and an Interaction Occurrence intersects each other

<table>
<thead>
<tr>
<th>Relation</th>
<th>Assessment</th>
<th>Rejection reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Combined Fragment contains an Interaction Occurrence</td>
<td>Acceptable</td>
<td>-</td>
</tr>
<tr>
<td>A Combined Fragment and an Interaction Occurrence outside of each other</td>
<td>Acceptable</td>
<td>-</td>
</tr>
<tr>
<td>A Combined Fragment inside an Interaction Occurrence</td>
<td>Unacceptable</td>
<td>Readability reducing</td>
</tr>
<tr>
<td>A Combined Fragment and an Interaction Occurrence intersects each other</td>
<td>Unacceptable</td>
<td>Readability reducing</td>
</tr>
</tbody>
</table>

Table 5.4: Combined Fragment/Interaction Occurrence relations

The first relation is a Combined Fragment inside an Interaction Occurrence. That is the four corners of the Combined Fragment is inside the four corners of the Interaction Occurrence. See (a) of figure 5.8 on the facing page as an example. There are no constraints in the specification forbidding this. But because this is rather meaningless we will consider this a readability reducing relation.

The second relation is a Combined Fragment containing an Interaction Occurrence. That is an Interaction Occurrence can be enclosed by one of the operands of the Combined Fragments. If the Interaction Occurrences four corners are inside the operands four corners the Interaction Occurrence is inside the Combined Fragment. See (b) of figure 5.8 on the preceding page. The specification does not have any constraints regarding this. It is both meaningful and readable so this can be viewed as acceptable.

The third relation is a Combined Fragment outside of an Interaction Occurrence as in (c) of figure 5.8 on the facing page. It is outside if they do not contain each other. The outside relation is defined more precisely under the Combined Fragment part.

And the fourth and last relation is a Combined Fragment and an Interaction Occurrence intersecting as in (d) of figure 5.8 on the preceding page. A Combined Fragment and an Interaction Occurrence are also intersecting if the Interaction Occurrence is located inside two or more of the Combined Fragments operands. There are no constraints in the specification prohibiting this. But the readability is clearly diminished. We will therefore suggest this be a readability reducing relation.
The discussion about acceptable/unacceptable relations between Combined Fragments and Interaction Occurrences resulted in two acceptable relations and two unacceptable relations. See table 5.4 on the preceding page for the complete result.

5.3 Combined Fragment editing behavior

This section will discuss how different editing operations for Combined Fragments should behave. The different editing operations are viewed separate. It will be discussed how an editing operation can affect other elements. If an edit results in an unacceptable relation between elements there will be given a method to solve this. The solutions given are only examples. They are given to show that it is possible to resolve the conflict. Section 5.2 on page 36 defines acceptable and unacceptable relations between Combined Fragments and other elements.

Before we start looking at some of the different editing operations used on Combined Fragments we will present some principles. These are to help guide the editing behavior.

The first principle is elements inside a Combined Fragment will, if possible, stay inside after the edit. This is because we view elements inside a Combined Fragment as “owned” by the Combined Fragment.

The next principle is connected to the modification of diagrams. If an unacceptable relation is created from an edit there must be performed some modification so the unacceptable relation is turned into an acceptable relation. We will try to use the expansion operations defined under 5.1 on page 31 to correct the errors.

The third principle takes effect if expansion does not solve the conflict. Then we will recommend to undo the edit operation. If this is done the diagram is unaffected by the edit.

5.3.1 Vertical move of Combined Fragment

Vertical movement of Combined Fragments is discussed here. We will look at the different effects it can have on other elements and draw a conclusion from this discussion. In the conclusion it will be Combined Fragment relations are potentially difficult to handle.

Messages
We are working from the assumption that the diagram the edit will be performed on is correct. From the discussion about acceptable/unacceptable
relations (see section 5.2 on page 36) we got four different acceptable relations between Messages and Combined Fragments:

1. A Message inside a Combined Fragment and not connected to the Combined Fragment

2. A Message inside a Combined Fragment and connected to the Combined Fragment

3. A Message outside a Combined Fragment and not connected to the Combined Fragment

4. A Message outside a Combined Fragment and connected to the Combined Fragment

The unacceptable relations between Combined Fragments and Messages are:

A. A Message connected to and intersecting a Combined Fragment

B. A Message not connected to and intersecting a Combined Fragment

We will see if a vertical move of a Combined Fragment can turn an acceptable relation into an unacceptable relation. If the edit can result in an unacceptable relation we will try to perform a form of expansion to get an acceptable relation. If this is not possible we will undo the edit. An undo will restore the diagram to the acceptable state it was in before the edit took place.

The first relation is a Message inside a Combined Fragment without being connected to the Combined Fragment. We will use the principle that elements inside a Combined Fragment should stay inside the Combined Fragment after an edit of the Combined Fragment. Letting the Message have the same location relative to the Combined Fragment before and after the vertical move will see to that the relation between the Message and the Combined Fragment is the same as before the vertical move, and therefore acceptable.

The second relation is a Message inside and connected to the Combined Fragment. This relation will have the same behavior as the first relation and not lead to any problems.

The third relation is a Message that is outside and not connected to a Combined Fragment. An example of this relation can be seen in (a) of figure 5.9 on the following page. After a vertical move of the Combined Fragment the Message can be in unacceptable relation B. This is unacceptable and we recommend therefore to turn the unacceptable relation into an
Figure 5.9: Combined Fragment in relation with Message - 3rd relation
acceptable relation. Unacceptable relation B can be viewed in (b) of figure 5.9.

But which acceptable relation should this unacceptable relation be turned into? It would be strange to start connecting the Message to the Combined Fragment after a move of the Combined Fragment. This would mean removing one of the other connections the Message has and replace it with a connection to the Combined Fragment. Therefore the two Combined Fragment/Message relations 2 and 4, where the Message is connected to the Combined Fragment are not options. This leaves the two not connected relations. The inside not connected relation can also create problems. There is no guarantee the Combined Fragment is big enough or covers the same area as the Message. Fixing a relation as the one seen in (b) of figure 5.9 on the preceding page by turning it into the inside relation would mean enlarging the Combined Fragment. But that seems a bit excessive. Therefore we have one acceptable relation left. This is the outside not connected relation.

It would be best for both elements to have the same relation to Lifelines before and after the modification. Any modification should therefore be done in the vertical direction.

To go from the intersect relation to the outside not connected relation it must first be decided if the Message should be placed over or under the Combined Fragment. This can be done by finding the top point of the Message and compare this point to the top and bottom of the Combined Fragment. If the top point of the Message is closer to the top of the Combined Fragment the Messages is placed over the Combined Fragment. And if the point is closer to the bottom of the Combined Fragment the Messages is placed under the Combined Fragment. When this is decided it will be possible to perform a vertical expansion. The vertical expansion operation would have to be modified slightly.

In the case of the Message going over the Combined Fragment the vertical expansion should be performed right above the Combined Fragment. The vertical expansion must not affect the Message. This will lead to that the Combined Fragment is pushed down-wards and eventually the Combined Fragment and the Message will no longer intersect. An example of this can be seen in figure 5.9 on the facing page. In (c) the vertical expansion line is introduced right above the Combined Fragment. In (d) the expansion has been performed, but observe that the Message is not affected by the expansion. And the relation between the Combined Fragment and the Message is again acceptable.

If the Message is to be placed under the Combined Fragment the vertical expansion must be performed in a different way. In this case the vertical expansion should be performed right above the Message. The vertical expansion
must not affect the Combined Fragment or any of the elements contained in the Combined Fragment. This will lead to that the Message is pushed downwards and eventually the Combined Fragment and the Message will no longer intersect.

We will call this special variation of the vertical expansion operation for *separating vertical expansion*. Instead of letting this operation only be limited to remove intersection of Combined Fragments and Messages we can say that it can work in the same way for all elements. The procedure would be the same. If there are two elements in an unacceptable relation we will turn the unacceptable relation into an outside relation. To do this we start by finding out which element should be under the other. Then we perform a vertical expansion pushing the lower element down-wards, at the same time the other element is not affected by the expansion. The two elements will no longer be in an unacceptable relation and they are outside of each other.

The third relation can therefore lead to the unacceptable relation B, but this could be fixed with the separating vertical expansion operation.

The fourth relation is Messages outside and connected to a Combined Fragment. This relation can be seen in figure 5.10 on the next page. Vertical movement of a Combined Fragment with this relation will not change the relation. The relation will still be the same. See (b) and (c) of figure 5.10 to get an idea of how the Message will behave.

To summarize the discussion about vertical movement of Combined Fragments with Message relations, we can look briefly at the four acceptable relations.

- The first relation will not create problems.
- The second relation will not create problems.
- The third relation can lead to the unacceptable relation B, but using the separating vertical expansion operation will solve this problem.
- The fourth relation will not create problems.

**Lifelines**

There are no unacceptable relations between Combined Fragments and Lifelines, we will therefore not discuss Combined Fragment/Lifeline relations.

**Combined Fragments**

If the diagram is without unacceptable relations a Combined Fragment can be related to another Combined Fragment in one of the following three ways:

1. A Combined Fragment inside a Combined Fragment
2. A Combined Fragment contains a Combined Fragment
(a) Outside connected relation

(b) After vertical move of Combined Fragment up-wards

(c) After vertical move of Combined Fragment down-wards

Figure 5.10: Combined Fragment in relation with Message - 4th relation
3. Two Combined Fragments outside of each other

There is one unacceptable relation between Combined Fragments:

A. Two Combined Fragment intersects each other

We will look at the three acceptable relations separately. We will see if a vertical move of a Combined Fragment can turn an acceptable relation into an unacceptable relation.

The first relation is the inside relation. A vertical move of the contained Combined Fragment can result in three different relations. The contained Combined Fragment can still be inside the Combined Fragment, the two Combined Fragments can be outside of each other or the two Combined Fragments can intersect. It is not possible to vertically move a contained Combined Fragment and end up in the situation that the contained Combined Fragment becomes the container. But we could end up in the situation where the two Combined Fragments share relation A. See section 5.2.3 on page 41 for the definition of intersecting Combined Fragments. (b) of figure 5.11 shows this relation. This should be fixed somehow. We can make use of the separating vertical expansion. This will take relation A and turn it into relation three.

The second relation is a Combined Fragment containing another Combined Fragment. Just like Combined Fragments containing Messages we will move the “owned” Combined Fragment with the container. The relation will be unchanged and therefore acceptable.

The third relation is two Combined Fragments not containing each other, the so called outside relation. After moving one of the Combined Fragments...
there are four possible situations. The moved Combined Fragment is either containing or inside the other Combined Fragment, they are still outside or they intersect. The first three relations are the acceptable relations so they are OK. But the intersect scenario must be dealt with. We can actually use the separating vertical expansion on this situation too.

To summarize the discussion about vertical movement of Combined Fragments with Combined Fragment relations, we can look briefly at the three acceptable relations.

The third relation can lead to unacceptable relation A, but it can be fixed with the separating vertical expansion operation.

The second relation will not create problems.

The third relation can lead to unacceptable relation A, but it can be fixed with the separating vertical expansion operation.

Interaction Occurrences
If the diagram is without unacceptable relations a Combined Fragment can be related to another Combined Fragment in one of the following two ways:

1. A Combined Fragment contains an Interaction Occurrence
2. A Combined Fragment and an Interaction Occurrence outside of each other

There are two unacceptable relations between Combined Fragments and Interaction Occurrences:

A. A Combined Fragment inside an Interaction Occurrence
B. A Combined Fragment and an Interaction Occurrence intersects each other

We will look at the two acceptable relations separately. We will see if a vertical move of a Combined Fragment can turn an acceptable relation into an unacceptable relation.

The first relation is a Combined Fragment containing an Interaction Occurrence. We will follow the principle that elements inside Combined Fragments should stay inside the Combined Fragment after the edit. The Interaction Occurrence will therefore be moved along with the Combined Fragment. This will preserve the acceptable relation and will not create any problems.

The second relation is a Combined Fragment and an Interaction Occurrence being outside of each other. In the event of this relation there can be
several results from a vertical move of the Combined Fragment. After the move the Combined Fragment can contain the Interaction Occurrence, they can still be outside of each other or the Interaction Occurrence intersects or contains the Combined Fragment. The two first situations are acceptable and do not need to be fixed. In the case of intersecting or an Interaction Occurrence containing a Combined Fragment we have unacceptable relations A and B. But again we can use the separating vertical expansion operation. An example of this can be seen in figure 5.12 on the facing page.

To summarize the discussion about vertical movement of Combined Fragments with Interaction Occurrence relations, we can look at the two acceptable relations.

The first relation will not create problems.

The second relation can lead to unacceptable relations A and B, but these can be fixed with the separating vertical expansion operation.

5.3.2 Horizontal move of Combined Fragment

Messages
There are four acceptable relations between Combined Fragments and Messages:

1. A Message inside a Combined Fragment and not connected to the Combined Fragment
2. A Message inside a Combined Fragment and connected to the Combined Fragment
3. A Message outside a Combined Fragment and not connected to the Combined Fragment
4. A Message outside a Combined Fragment and connected to the Combined Fragment

The unacceptable relations between Combined Fragments and Messages are:

A. A Message connected to and intersecting a Combined Fragment
B. A Message not connected to and intersecting a Combined Fragment

We will look at what can happen with each of these four relations after a horizontal move of the Combined Fragment. If the move results in an unacceptable relation we can try to fix it by using one of the expansion operations...
Figure 5.12: Vertical move of Combined Fragment with Interaction Occurrence - 2nd relation
operations. If this does not work we will recommend horizontal move be undone.

The first relation, a Message inside, but not connected to a Combined Fragment. This means the Message is connected to two elements inside the Combined Fragment. Moving the Combined Fragment should not move Lifelines. Therefore Messages connected to Lifelines can be a problem. See figure 5.13 for a before and after look at this situation. There the Message intersects the Combined Fragment. This is the unacceptable relation B. If the situation is like in (b) of figure 5.13 we will suggest making use of the separating vertical expansion operation. This will lead to the unacceptable relation B turning into the third acceptable relation.

The second relation is a Message inside and connected to a Combined Fragment. This relation can also create problems. An example can be seen in figure 5.14 on the next page. And as seen in (b), we can get a transition from an acceptable relation to the unacceptable relation A. This is not wanted. Trying to fix this we can use expansion, but this will not turn this relation into an acceptable relation. We will therefore recommend undoing any horizontal move resulting in such a relation.

The third relation is Messages outside and not connected to a Combined Fragment. The relation can be viewed in figure 5.15 on the facing page. In (b) horizontal move of a Combined Fragment results in the unacceptable relation B. The Message intersects the Combined Fragment. If this happens we will suggest making use of the separating vertical expansion operation. This will turn unacceptable relation B into the third acceptable relation.

The fourth relation is a Message outside and connected to a Combined Fragment. See figure 5.16 on page 58 for an example. This relation can
Figure 5.14: Horizontal move of Combined Fragment with a Message - 2nd relation

Figure 5.15: Horizontal move of Combined Fragment with a Message - 3rd relation
also result in the Message intersecting the Combined Fragment (see (b)) after a horizontal move of the Combined Fragment. This is the unacceptable relation A. We will recommend trying to fix this using the separating vertical expansion operation, but it will not work on this relation because the Message will still be connected. We will therefore suggest to undo any move resulting in this relation.

To summarize the discussion about horizontal movement of Combined Fragments with Message relations, we can look briefly at the four acceptable relations.

The first relation can lead to unacceptable relation B, but it can be fixed with the separating vertical expansion operation.

The second relation can lead to unacceptable relation A. This is not fixable with expansion operations. We will therefore suggest to undo the edit operations resulting in this.

The third can lead to unacceptable relation B, but it can be fixed with the separating vertical expansion operation.

The fourth relation can lead to unacceptable relation A. This is not fixable with expansion operations. We will therefore suggest to undo the edit operations resulting in this.
Lifelines
There are no unacceptable relations between Combined Fragments and Lifelines, therefore we will not discuss Combined Fragment/Lifeline relations.

Combined Fragments
If the diagram is without unacceptable relations a Combined Fragment can be related to another Combined Fragment in one of the following three ways:

1. A Combined Fragment inside a Combined Fragment
2. A Combined Fragment contains a Combined Fragment
3. Two Combined Fragments outside of each other

There is one unacceptable relation between Combined Fragments:

A. Two Combined Fragment intersects each other

We will look at the three acceptable relations separately. We will see if a horizontal move of a Combined Fragment can turn an acceptable relation into an unacceptable relation.

The first relation is a Combined Fragment inside another Combined Fragment. A horizontal move of the contained Combined Fragment can result in the unacceptable relation A. An example can be viewed in figure 5.17 on the following page. In this situation we will recommend to perform the separating vertical expansion operation to separate the two Combined Fragments.

The second relation is a Combined Fragment containing another Combined Fragment. We will apply the principle of letting elements inside a Combined Fragment move along with the Combined Fragment. The inner Combined Fragment will therefore have the same relation before and after the move. This causes no problems.

The third and last relation is Combined Fragments outside of each other. See figure 5.18 on the next page for an example. Here the neg Combined Fragment is moved. This can result in intersecting Combined Fragments (as seen in (b)). This is the unacceptable relation A, so we recommend to fix it so that it becomes an acceptable relation. We can use the same approach outlined in the first relation. There we suggested to use the separating vertical expansion operation. This fixes the problem.

To summarize the discussion about horizontal movement of Combined Fragments with Combined Fragment relations, we can look briefly at the three acceptable relations.
Figure 5.17: Horizontal move of Combined Fragment in relation with a Combined Fragment - 1st relation

Figure 5.18: Horizontal move of Combined Fragment in relation with a Combined Fragment - 3rd relation
The first relation can lead to unacceptable relation A, but it can be fixed with the separating vertical expansion operation. The second relation will not create problems. The third relation can lead to unacceptable relation A, but it can be fixed with the separating vertical expansion operation.

**Interaction Occurrences**

If the diagram is without unacceptable relations a Combined Fragment can be related to another Combined Fragment in one of the following two ways:

1. A Combined Fragment contains an Interaction Occurrence
2. A Combined Fragment and an Interaction Occurrence outside of each other

There are two unacceptable relations between Combined Fragments and Interaction Occurrences:

A. A Combined Fragment inside an Interaction Occurrence
B. A Combined Fragment and an Interaction Occurrence intersects each other

We will look at the two acceptable relations separately. We will see if a horizontal move of a Combined Fragment can turn an acceptable relation into an unacceptable relation.

The first relation is a Combined Fragment containing an Interaction Occurrence. Again we use the principal that the elements inside a Combined Fragment will be moved along with the Combined Fragment. This means the Interaction Occurrence will have the same relation after the move. This will not create any problems.

The second relation is a Combined Fragment and an Interaction Occurrence outside of each other. This is a more complicated situation. See figure 5.19 on the following page for an example of this relation turning into the unacceptable relation B. This relation can also end in the other unacceptable relation A. We recommend to turn both unacceptable relation into acceptable with the separating vertical expansion operation. This will remove the unacceptable relations.

To summarize the discussion about horizontal movement of Combined Fragments with Interaction Occurrence relations, we will look briefly at the two acceptable relations.
Figure 5.19: Horizontal move of Combined Fragment in relation with an Interaction Occurrence - 2nd relation
The first relation will not create problems. The second can lead to both unacceptable relations A and B, but these can be fixed with the separating vertical expansion operation.

5.3.3 Vertical scale of Combined Fragment

This part will deal with vertical scale of Combined Fragments. The scale can be split into two incidents, scale to make the Combined Fragment larger or to scale to make the Combined Fragment smaller. We will call the two different scales for \textit{upsize-scale} and \textit{downsize-scale} respectively.

Messages

There are four acceptable Combined Fragment/Message relations:

1. A Message inside a Combined Fragment and not connected to the Combined Fragment
2. A Message inside a Combined Fragment and connected to the Combined Fragment
3. A Message outside a Combined Fragment and not connected to the Combined Fragment
4. A Message outside a Combined Fragment and connected to the Combined Fragment

And two unacceptable relations:

A. A Message connected to and intersecting a Combined Fragment
B. A Message not connected to and intersecting a Combined Fragment

We are looking for acceptable relations turning into unacceptable after a vertical Combined Fragment scale.

The first relation is a Message inside, but not connected to a Combined Fragment. When it comes to the vertical upsize-scale we will get no problems. The Message will still be inside, the only thing that happens is that the Message will have more room inside the Combined Fragment. The other possibility is vertical downsize-scale. See figure 5.20 on the next page for an example of this. Here we get the unacceptable relation B as a result of the scaling. We suggest to use separating vertical expansion on this relation. This will turn the unacceptable relation B into the third acceptable relation.

The second relation is a Message inside and connected to a Combined Fragment. We will start to look at the vertical upsize-scale. This will have
Figure 5.20: Vertical scale of Combined Fragment in relation with a Message - 1st relation

no effect on the Message. The Message is connected to the vertical sides of the Combined Fragment and the vertical sides are the ones increasing during a vertical upsize-scale. Besides from that this is the same as a vertical upsize-scale for Messages inside and not connected. Now we will look at vertical downsize-scale. It is possible to get the unacceptable relation A. There is also another possible error. See figure 5.21 on the facing page to see this error. The Message is connected to the Combined Fragment, but the scale removes this point. This gives something we will consider an unacceptable relation, and trying to fix these relations using the vertical expansion operation will not work. We will therefore recommend undoing edits leading to these situations.

The third relation is a Message outside and not connected to a Combined Fragment. We will start to look at vertical upsize-scale. See figure 5.22 on the next page for an example of this. The result is a Message intersecting the Combined Fragment. This is unacceptable relation B. It is possible to use the separating vertical expansion operation to make room between the Message and the Combined Fragment. This will allow the Combined Fragment to be scaled and at the same time let the Message be above or under the Combined Fragment. The vertical downsize-scale will not create any problems for a Message outside and not connected to a Combined Fragment.

The fourth and last relation is a Message outside and connected to a Combined Fragment. We will start to look at vertical upsize-scale. The Message is connected to one of the vertical sides of the Combined Fragment and the vertical sides are the ones increasing during a vertical upsize-scale. This means the relation will be intact. When it comes to the vertical downsize-scale it is possible to go from the outside and connected relation to the outside and not
Figure 5.21: Vertical scale of Combined Fragment in relation with a Message - 2nd relation

Figure 5.22: Vertical scale of Combined Fragment in relation with a Message - 3rd relation
connected relation. See figure 5.23. These are both acceptable according to
the discussion about Message/Combined Fragment relations, but a Message
needs two elements to connect to. We will therefore suggest this relations
is viewed as unacceptable. Trying to solve this problem with the vertical
expansion operation will not work. We will therefore recommended this edit
be undone.

To summarize the discussion about vertical scale of Combined Fragments
with Message relations, we can look briefly at the four acceptable relations.
The first relation can lead to unacceptable relation B, but it can be fixed
with the separating vertical expansion operation.
The second relation can lead to both unacceptable relation A and an
undefined error. These are not fixable with expansion operations. We will
therefore suggest to undo the edit operations resulting in this.
The third relation can lead to unacceptable relation B, but it can be fixed
with the separating vertical expansion operation.
The fourth relation can end in an undefined unacceptable relation not
fixable with expansion operations. We will therefore suggest to undo the
edit operations resulting in this.

**Lifelines**
There are no unacceptable relations between Combined Fragments and Life-
lines, therefore we will not discuss Combined Fragment/Lifeline relations.

**Combined Fragments**
If the diagram is without unacceptable relations a Combined Fragment can
be related to another Combined Fragment in one of the following three ways:
1. A Combined Fragment inside a Combined Fragment

2. A Combined Fragment contains a Combined Fragment

3. Two Combined Fragments outside of each other

There is one unacceptable relation between Combined Fragments:

A. Two Combined Fragment intersects each other

We will look at the three acceptable relations separately. We will see if a vertical scale of a Combined Fragment can turn an acceptable relation into an unacceptable relation.

The first relation is a Combined Fragment inside another Combined Fragment. See (a) of figure 5.24 on the next page. Here the neg Combined Fragment is the one we will scale. If we perform a vertical upsize-scale of the neg Combined Fragment we get the situation seen in (b). This is the unacceptable relation A, where Combined Fragments intersects each other. In this situation we will recommend to use the separating vertical expansion operation. This will fix the unacceptable relation. See (c) of figure 5.24 on the following page for the result of this on (b) of the same figure. The vertical downsize-scale will not create any problems.

The second relation is a Combined Fragment containing another Combined Fragment. See (a) of figure 5.25 on page 69. Here the alt Combined Fragment is the one we will scale. A vertical upsize-scale will preserve the acceptable relation. But the vertical downsize-scale of the alt Combined Fragment will cause a problem. This can be seen in (b) of figure 5.25 on page 69. Here the two Combined Fragments intersect, which is unacceptable relation A. We suggest using the separating vertical expansion operation so the two Combined Fragments become outside of each other. See (c) of figure 5.25 for the result of doing this on (b).

The third and last relation is Combined Fragments outside of each other. A vertical upsize-scale of one of the Combined Fragments can result in relation A. An easy way of resolving this is to use the same approach as on the other two relations. The vertical downsize-scale of Combined Fragments outside of each other will not change the relation.

To summarize the discussion about vertical scale of Combined Fragments with Combined Fragment relations, we can look briefly at the three acceptable relations.

The first relation can lead to unacceptable relation A, but it can be fixed with the separating vertical expansion operation.

The second relation can lead to unacceptable relation A, but it can be fixed with the separating vertical expansion operation.
Figure 5.24: Vertical scale of Combined Fragment in relation with a Combined Fragment - 1st relation
Figure 5.25: Vertical scale of Combined Fragment in relation with a Combined Fragment - 2nd relation
The third relation can lead to unacceptable relation A, but it can be fixed with the separating vertical expansion operation.

**Interaction Occurrences**

If the diagram is without unacceptable relations a Combined Fragment can be related to another Combined Fragment in one of the following two ways:

1. A Combined Fragment contains an Interaction Occurrence
2. A Combined Fragment and an Interaction Occurrence outside of each other

There are two unacceptable relations between Combined Fragments and Interaction Occurrences:

A. A Combined Fragment inside an Interaction Occurrence
B. A Combined Fragment and an Interaction Occurrence intersects each other

We will look at the two acceptable relations separately. We will see if a vertical scale of a Combined Fragment can turn an acceptable relation into an unacceptable relation. As before we will decompose the vertical scale into vertical upsize-scale and vertical downsize-scale.

The first relation is a Combined Fragment containing an Interaction Occurrence. See (a) of figure 5.26 on the next page for an example of this relation. The vertical upsize-scale of the Combined Fragment will not lead to any problems. The vertical downsize-scale can end in unacceptable relation B. See (b) as an example. We recommend using the separating vertical expansion operation to solve this.

The second relation is a Combined Fragment and an Interaction Occurrence outside of each other. See (a) of 5.27 on the facing page for an example of this. We will start with vertical upsize-scale of the Combined Fragment. This can end up with relation B. This can be viewed in (b) of figure 5.27. This is unacceptable and must be dealt with. This can be solved in the same way as before. Combined Fragments and Interaction Occurrence with the intersect relation will therefore be changed to have the outside relation with use of the separating vertical expansion operation. The vertical downsize-scale will not create any problems.

To summarize the discussion about vertical scale of Combined Fragments with Combined Fragment relations, we can look briefly at the two acceptable relations.
Figure 5.26: Vertical scale of Combined Fragment in relation with an Interaction Occurrence - 1st relation

Figure 5.27: Vertical scale of Combined Fragment in relation with an Interaction Occurrence - 2nd relation
The first relation can create an unacceptable relation B, but it can be fixed with the separating vertical expansion operation.

The second relation can create an unacceptable relation B, but it can be fixed with the separating vertical expansion operation.

5.3.4 Horizontal scale of Combined Fragment

Horizontal scale can be split into two incidents, scale to make the Combined Fragment larger or scale to make the Combined Fragment smaller. We will call the two different scales for *upsize-scale* and *downsize-scale* respectively.

Messages
There are four acceptable Combined Fragment/Message relations:

1. A Message inside a Combined Fragment and not connected to the Combined Fragment
2. A Message inside a Combined Fragment and connected to the Combined Fragment
3. A Message outside a Combined Fragment and not connected to the Combined Fragment
4. A Message outside a Combined Fragment and connected to the Combined Fragment

And two unacceptable relations:

A. A Message connected to and intersecting a Combined Fragment
B. A Message not connected to and intersecting a Combined Fragment

We are looking for acceptable relations turning into unacceptable after a horizontal Combined Fragment scale.

The first relation is a Message inside, but not connected to a Combined Fragment. When it comes to the horizontal upsize-scale we will get no problems. The Message will still be inside, the only thing that happens is that the Message will have more room inside the Combined Fragment. The other possibility is horizontal downsize-scale. See figure 5.28 on the next page for an example of this. Here we get unacceptable relation B as a result of the scaling. It helps to perform separating vertical expansion on this relation, so we will suggest using it to resolve the conflict. This will remove the unacceptable relation.
The second relation is a Message inside and connected to a Combined Fragment. We will start to look at the horizontal upsize-scale. This will have no effect on the relation. Now we will look at vertical downsize-scale. It is possible to get the unacceptable relation A. See figure 5.29 on the following page to see this error. The Message is connected to the Combined Fragment, but the scale causes the Lifeline the Message is connected to to be outside of the Combined Fragment. Therefore the Message is both inside and outside at the same time, creating an intersection of the Combined Fragment. This is unacceptable relation A, and trying to fix it using the expansion operations will not work. Therefore we will recommend to undo this edit.

The third relation is a Message outside and not connected to a Combined Fragment. We will start to look at horizontal upsize-scale. See figure 5.30 on the next page for an example of this. The result is a Message intersecting the Combined Fragment. This is unacceptable relation B. It helps to perform separating vertical expansion on this relation, so we will recommend it. This will solve the conflict. When it comes to the horizontal downsize-scale it will not change the relation.

The fourth and last relation is a Message outside and connected to a Combined Fragment. We will start to look at horizontal upsize-scale. See figure 5.31 on page 75. Here we have a skewed Message intersecting the Combined Fragment after a horizontal scale of the Combined Fragment. This is the unacceptable relation A and trying to fix this error with the expansion operations will not work. Therefore we will recommend to undo this edit. When it comes to the horizontal downsize-scale it will not change the relation. The Message will still be outside and connected to the Combined Fragment.
Figure 5.29: Horizontal scale of Combined Fragment in relation with a Message - 2nd relation

Figure 5.30: Horizontal scale of Combined Fragment in relation with a Message - 3rd relation
To summarize the discussion about horizontal scale of Combined Fragments with Message relations, we can look briefly at the four acceptable relations.

The first relation can lead to unacceptable relation B, but it can be fixed with the separating vertical expansion operation.

The second relation can lead to unacceptable relation A, not fixable with expansion operations. We will therefore suggest to undo the edit operations resulting in this relation.

The third relation can lead to unacceptable relation B, but it can be fixed with the separating vertical expansion operation.

The fourth relation can lead to unacceptable relation A, not fixable with expansion operations. We will therefore suggest to undo the edit operations resulting in this relation.

Lifelines
There are no unacceptable relations between Combined Fragments and Lifelines, therefore we will not discuss Combined Fragment/Lifeline relations.

Combined Fragments
If the diagram is without unacceptable relations a Combined Fragment can be related to another Combined Fragment in one of the following three ways:

1. A Combined Fragment inside a Combined Fragment

2. A Combined Fragment contains a Combined Fragment
3. Two Combined Fragments outside of each other

There is one unacceptable relation between Combined Fragments:

A. Two Combined Fragment intersects each other

We will look at the three acceptable relations separately. We will see if a horizontal scale of a Combined Fragment can turn an acceptable relation into an unacceptable relation.

The first relation is a Combined Fragment inside another Combined Fragment. This can be seen in (a) of figure 5.32 on the next page. Here the Combined Fragment is the one we will perform the horizontal scale on. A horizontal upsize-scale of the contained Combined Fragment can lead to relation A. See (b) for an example of this. Again we will suggest the use of the separating vertical expansion operation. This means placing the Combined Fragments outside of each other. This is done on (b) and the result can be seen in (c) of figure 5.32 on the facing page. The horizontal downsize-scale of a Combined Fragment will not cause any problems for this relation.

The second relation is a Combined Fragment containing another Combined Fragment. A horizontal upsize-scale of the containing Combined Fragment will not lead to relation A. But a horizontal downsize-scale can lead to that. Again we will recommend using separating vertical expansion to solve this unacceptable relation.

The third and last relation is Combined Fragments outside of each other. See (a) of figure 5.33 on page 78 for an example of this. A horizontal upsize-scale of one of the Combined Fragments can lead to relation A. This is shown in (b). We will suggest to do the same as always and use separating vertical expansion. The result on (b) can be seen in (c).

To summarize the discussion about horizontal scale of Combined Fragments with Combined Fragment relations, we can look briefly at the three acceptable relations.

The first relation can lead to unacceptable relation A, but it can be fixed with the separating vertical expansion operation.

The second relation can lead to unacceptable relation A, but it can be fixed with the separating vertical expansion operation.

The third relation can lead to unacceptable relation A, but it can be fixed with the separating vertical expansion operation.

Interaction Occurrences

If the diagram is without unacceptable relations a Combined Fragment can be related to another Combined Fragment in one of the following two ways:

1. A Combined Fragment contains an Interaction Occurrence
Figure 5.32: Horizontal scale of Combined Fragment in relation with a Combined Fragment - 1st relation
Figure 5.33: Horizontal scale of Combined Fragment in relation with a Combined Fragment - 3rd relation
2. A Combined Fragment and an Interaction Occurrence outside of each other

There are two unacceptable relations between Combined Fragments and Interaction Occurrences:

A. A Combined Fragment inside an Interaction Occurrence

B. A Combined Fragment and an Interaction Occurrence intersects each other

We will look at the two acceptable relations separately. We will see if a vertical scale of a Combined Fragment can turn an acceptable relation into an unacceptable relation. As before we will decompose the horizontal scale into horizontal upsize-scale and horizontal downsize-scale.

The first relation is a Combined Fragment containing an Interaction Occurrence. An example can be seen in (a) of figure 5.34 on the following page. When it comes to the horizontal upsize-scale of the Combined Fragment we get no problems. But a horizontal downsize-scale of the Combined Fragment can end in unacceptable relation B. See (b). We suggest separating vertical expansion to fix this. This can be seen in (c).

The second relation is a Combined Fragment and an Interaction Occurrence outside of each other. This is about the same discussion as the previous, but the horizontal upsize-scale causes the problem. See (a), (b) and (c) of figure 5.35 on page 81 to see before the scale, after the scale and after the fix respectively. The horizontal downsize-scale of the Combined Fragment will not create any problems.

To summarize the discussion about horizontal scale of Combined Fragments with Combined Fragment relations, we can look briefly at the two acceptable relations.

The first relation can lead to unacceptable relation B, but it can be fixed with the separating vertical expansion operation.

The second relation can lead to unacceptable relation B, but it can be fixed with the separating vertical expansion operation.

5.3.5 Summary of Combined Fragment editing behavior

There are five unacceptable relations between Combined Fragments and other elements. These are shown in table 5.5 on page 82.

In table 5.6 on page 82 we summarize the effect of the four edit operations discussed for Combined Fragments. The table shows which unacceptable relation the different editing operations can lead to. The basis for the discussion
Figure 5.34: Horizontal scale of Combined Fragment with an Interaction Occurrence - 1st relation
Figure 5.35: Horizontal scale of Combined Fragment with an Interaction Occurrence - 2nd relation
### Table 5.5: Summary of unacceptable Combined Fragment relations

<table>
<thead>
<tr>
<th>Element</th>
<th>Violating relation</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message</td>
<td>A Message connected to and intersecting a Combined Fragment</td>
<td>MSG1</td>
</tr>
<tr>
<td></td>
<td>A Message not connected to and intersecting a Combined Fragment</td>
<td>MSG2</td>
</tr>
<tr>
<td>Combined Fragment</td>
<td>Two Combined Fragment intersects each other</td>
<td>CF1</td>
</tr>
<tr>
<td>Interaction Occurrence</td>
<td>A Combined Fragment inside an Interaction Occurrence</td>
<td>IO1</td>
</tr>
<tr>
<td></td>
<td>A Combined Fragment and an Interaction Occurrence intersects each other</td>
<td>IO2</td>
</tr>
</tbody>
</table>

### Table 5.6: Summary of Combined Fragment editing behavior

<table>
<thead>
<tr>
<th>Edit operation</th>
<th>Violating relation</th>
<th>Editor reaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vertical move</td>
<td>MSG2</td>
<td>Separating vertical expansion</td>
</tr>
<tr>
<td></td>
<td>CF1</td>
<td>Separating vertical expansion</td>
</tr>
<tr>
<td></td>
<td>IO1</td>
<td>Separating vertical expansion</td>
</tr>
<tr>
<td></td>
<td>IO2</td>
<td>Separating vertical expansion</td>
</tr>
<tr>
<td>Horizontal move</td>
<td>MSG1</td>
<td>Undo</td>
</tr>
<tr>
<td></td>
<td>MSG2</td>
<td>Separating vertical expansion</td>
</tr>
<tr>
<td></td>
<td>CF1</td>
<td>Separating vertical expansion</td>
</tr>
<tr>
<td></td>
<td>IO1</td>
<td>Separating vertical expansion</td>
</tr>
<tr>
<td></td>
<td>IO2</td>
<td>Separating vertical expansion</td>
</tr>
<tr>
<td>Vertical scale</td>
<td>MSG1</td>
<td>Undo</td>
</tr>
<tr>
<td></td>
<td>MSG2</td>
<td>Separating vertical expansion</td>
</tr>
<tr>
<td></td>
<td>CF1</td>
<td>Separating vertical expansion</td>
</tr>
<tr>
<td></td>
<td>IO2</td>
<td>Separating vertical expansion</td>
</tr>
<tr>
<td>Horizontal scale</td>
<td>MSG1</td>
<td>Undo</td>
</tr>
<tr>
<td></td>
<td>MSG2</td>
<td>Separating vertical expansion</td>
</tr>
<tr>
<td></td>
<td>CF1</td>
<td>Separating vertical expansion</td>
</tr>
<tr>
<td></td>
<td>IO2</td>
<td>Separating vertical expansion</td>
</tr>
</tbody>
</table>

Table 5.6: Summary of Combined Fragment editing behavior
was a legal relation. For example in the discussion about vertical move of Combined Fragments we have only discussed which unacceptable relations a vertical move of a Combined Fragment in acceptable relations can lead to. The table also summarizes how we recommended for the editor to react if an edit operation led to an unacceptable relation.

All five unacceptable relations shows up in table 5.6. This means we have proposed an editor reaction to all five relations. With use of the separating vertical expansion operation it is possible to resolve four of the five unacceptable relations. The last relation, a Message connected to and intersecting a Combined Fragment, is not as easily resolved. The suggested editor reaction is to undo the edit operation leading to this unacceptable relation.

An undefined error was found in the discussion about vertical scale of Combined Fragment in relation with a Message. This shows that it is possible to find new errors using this approach to analyze editing behavior. A good idea is therefore to perform the analysis in iterations. Even though the analysis shown is this thesis seems to only be performed in one pass it has actually gone through several iterations.

5.4 Acceptable/unacceptable Interaction Occurrence relations

To help in the discussion about editing behavior of Interaction Occurrences we look at the possible relations an Interaction Occurrence can have with other elements. We will decide what is acceptable and what is unacceptable.

To differentiate if the relation is unacceptable because of the specification or our additional constraints we will create two unacceptable categories. If the relation is unacceptable according to the specification it will be a specification breaking relation. If it is unacceptable because of one of our additional constraints we will refer to it as a readability reducing relation.

5.4.1 Messages

There are six different relations an Interaction Occurrence can have with a Message:

1. A Message inside an Interaction Occurrence and not connected to the Interaction Occurrence

2. A Message inside an Interaction Occurrence and connected to the Interaction Occurrence
Figure 5.36: Interaction Occurrence in relation with Message
3. A Message outside an Interaction Occurrence and not connected to the Interaction Occurrence

4. A Message outside an Interaction Occurrence and connected to the Interaction Occurrence

5. A Message connected to and intersecting an Interaction Occurrence

6. A Message not connected to and intersecting an Interaction Occurrence

We will look at these six relations to determine if they are acceptable or unacceptable.

The first relation is a Message inside an Interaction Occurrence and not connected to the Interaction Occurrence. An example can be viewed in (a) of figure 5.36 on the facing page. There are no constraints disallowing this in the specification. Allowing it may lead to Messages covering the name of the Interaction Occurrence. Therefore we will add a constraint saying Messages cannot be inside Interaction Occurrences. We will therefore recommend this to be a readability reducing relation.

The second relation is a Message inside and connected to an Interaction Occurrence in the form of a Gate. See (b) of figure 5.36 as an example. The specification does not explicitly forbid this, but we proposed a constraint saying Messages cannot be inside Interaction Occurrences. Therefore we will suggest this is an unacceptable readability reducing relation.

The third relation is a Message being outside and not connected to an Interaction Occurrence. An example can be seen in (c) of figure 5.36 on the preceding page. The specification gives no constraints on this relation. And such a relation is both meaningful and highly readable. We will therefore allow it.

The fourth relation is a Message outside and connected with an Interaction Occurrence in the form of a Gate. View (d) of figure 5.36 for an example. Again the specification does not give constraints disallowing this. It is both meaningful and easy to read this relation. We will therefore allow it.

The fifth relation is a Message connected to and intersecting an Interaction Occurrence. This is when the boundary of a Interaction Occurrence is crossed by a Message. An example of this relation can be seen in (e) of figure 5.36 on the facing page. There are no constraints disallowing this in the specification. It can definitely have meaning, but the readability of the diagram is diminished. We will therefore suggest this is a readability reducing relation.

The sixth relation is a Message not connected to and intersecting an Interaction Occurrence. This is when the boundary of an Interaction Occurrence is crossed by a Message. Examples of this relation can be seen in (f)
Table 5.7: Interaction Occurrence/Message relations

of figure 5.36 on page 84. There are no constraints disallowing this in the specification. It can definitely have meaning, but the readability of the diagram is diminished. We will therefore suggest this is a readability reducing relation.

To summarize all six relations are acceptable according to the specification, but after adding a few new constraints we got only two acceptable relations. See table 5.7 for the final result of the Interaction Occurrence/Message relation discussion.

### 5.4.2 Lifelines

Interaction Occurrences can have the following three relations to a Lifeline:

1. An Interaction Occurrence covering a Lifeline
2. An Interaction Occurrence not covering a Lifeline

3. An Interaction Occurrence intersecting or enclosing the head of the Lifeline

Because of the assumption that Combined Fragments always must be under Lifeline heads a Interaction Occurrence will never intersect or enclose the head of a Lifeline. We will therefore disregard the third relation.

This discussion will be the same as for Lifelines in relation to Combined Fragments. The specification did not have constraints disallowing the two relations between Lifelines and Combined Fragments and there are no constraints disallowing the two relations between Lifelines and Interaction Occurrences. And there are no reasons with regards to the readability to disallow this.

The result of Interaction Occurrence/Lifeline relations are shown in table 5.8.

5.4.3 Combined Fragments

Combined Fragment and Interaction Occurrence relations are discussed in the Combined Fragment section 5.2 on page 36. The result was two acceptable
5.4.4 Interaction Occurrences

Interaction Occurrences have four possible relations to other Interaction Occurrences:

1. An Interaction Occurrence inside an Interaction Occurrence
2. An Interaction Occurrence contains an Interaction Occurrence
3. Two Interaction Occurrences outside of each other

Figure 5.38: Interaction Occurrence in relation with Interaction Occurrence and two unacceptable relations. The result was presented in table 5.4 on page 45.
Table 5.9: Interaction Occurrence/Interaction Occurrence relations

4. Two Interaction Occurrences intersects each other

The first relation is an Interaction Occurrence enclosed in another Interaction Occurrence. See (a) of figure 5.38 on the facing page as an example. There are no constraints in the specification forbidding this. But because this is rather meaningless we will recommend this is a readability reducing relation.

The second relation is an Interaction Occurrence containing an Interaction Occurrence. See (a) of figure 5.38 on the preceding page. Just as the inside relation this is not prohibited in the specification, but it is rather meaningless and therefore we recommend this is a readability reducing relation.

The third relation is Interaction Occurrences not enclosing each other. An example can be viewed in (b) of figure 5.38 on the facing page. This is acceptable according to the specification. And this relation is easy to read, so we will allow it. Therefore the relation is acceptable.

The fourth and last relation is intersecting Interaction Occurrences. See (c) of figure 5.38 on the preceding page for an example. The specification does not deal with this relation, but the readability is clearly reduced. Therefore we will suggest this is a readability reducing relation.

To get a summary of this discussion see table 5.9.

5.5 Interaction Occurrence editing behavior

5.5.1 Vertical move of Interaction Occurrence

Messages
There are two acceptable Interaction Occurrence/Message relations:
Figure 5.39: Vertical move of Interaction Occurrence with Message - 1st relation
Figure 5.40: Vertical move of Interaction Occurrence with Message - 2nd relation
1. A Message outside an Interaction Occurrence and not connected to the Interaction Occurrence

2. A Message outside an Interaction Occurrence and connected to the Interaction Occurrence

And there are four unacceptable Interaction Occurrence/Message relations:

A. A Message inside an Interaction Occurrence and not connected to the Interaction Occurrence

B. A Message inside an Interaction Occurrence and connected to the Interaction Occurrence

C. A Message connected to and intersecting an Interaction Occurrence

D. A Message not connected to and intersecting an Interaction Occurrence

We will discuss vertical Interaction Occurrence move in an acceptable diagram and see if it can lead to an unacceptable diagram.

The first relation is a Message being outside and not connected to an Interaction Occurrence. An example can be seen in (a) of figure 5.39 on page 90. A vertical Interaction Occurrence move can result in the two unacceptable relations A and D. Relation A is shown in (b) of figure 5.39 on page 90. We suggest using the separating vertical expansion operation to solve this. In (c) of figure 5.39 on page 90 this procedure is performed on (b).

The second relation is a Message outside and connected with an Interaction Occurrence in the form of a Gate. (a) of figure 5.40 on the preceding page shows an example. There are no problems associated with vertical movement of an Interaction Occurrence with this Message relation. See (b) and (c) to get an idea of the effect of a vertical Interaction Occurrence move.

To summarize the discussion about vertical move of Interaction Occurrences with Message relations, we can look briefly at the two acceptable relations.

The first relation can lead to unacceptable relations A and D, but it can be fixed with the separating vertical expansion operation.

The second relation creates no problems.

Lifelines

There are no unacceptable relations between Interaction Occurrences and Lifelines, therefore we will not discuss Interaction Occurrence/Lifeline relations.
Combined Fragments

There are two acceptable Interaction Occurrence/Combined Fragment relations:

1. A Combined Fragment contains an Interaction Occurrence
2. A Combined Fragment and an Interaction Occurrence outside of each other

And there are two unacceptable relations:

A. A Combined Fragment inside an Interaction Occurrence
B. A Combined Fragment and an Interaction Occurrence intersects each other

We will see if the acceptable relations can be turned into unacceptable by a vertical Interaction Occurrence move.

The first acceptable relation is an Interaction Occurrence inside a Combined Fragment. View (a) of figure 5.41 on the next page for an example. A vertical Interaction Occurrence move can turn this relation into the unacceptable relation B. We will fix this in the same way as before. Finding out which element is the top element and push the other down using the separating vertical expansion. See (b) and (c) of figure 5.41 on the following page for the unacceptable diagram and the fixed diagram respectively.

The second acceptable relation is an Interaction Occurrence and a Combined Fragment being outside of each other. An example can be seen in (a) of figure 5.42 on page 95. This relation can turn into both unacceptable Interaction Occurrence/Combined Fragment relations A and B. This means an Interaction Occurrence can enclose a Combined Fragment or they can intersect. See (b) for the latter. Both unacceptable relations can be turned into acceptable by using the separating vertical expansion and pushing the lowest element underneath the upper element. (c) of figure 5.42 on page 95 show the fix of the unacceptable relation in (b).

To summarize the discussion about vertical move of Interaction Occurrences with Combined Fragment relations, we can look briefly at the two acceptable relations.

The first relation can lead to unacceptable relation B, but it can be fixed with the separating vertical expansion operation.

The second relation can lead to unacceptable relations A and B, but these can be fixed with the separating vertical expansion operation.
Figure 5.41: Vertical move of Interaction Occurrence with Combined Fragment - 1st relation
Figure 5.42: Vertical move of Interaction Occurrence with Combined Fragment - 2nd relation
**Interaction Occurrences**

There is one acceptable relation between Interaction Occurrences:

1. Two Interaction Occurrences outside of each other

And three unacceptable relations:

A. An Interaction Occurrence inside an Interaction Occurrence

B. An Interaction Occurrence contains an Interaction Occurrence

C. Two Interaction Occurrences intersects each other

We are looking for situations where the acceptable relation becomes an unacceptable relation because of a vertical Interaction Occurrence move.

The first and only acceptable relation is Interaction Occurrences outside of each other. An example can be seen in (a) of figure 5.43 on the next page.

A vertical move can result in all three unacceptable relations. But we will suggest solving these errors in the same way we solved the problems connected with vertical move of Interaction Occurrences in relation with Combined Fragment. We suggest using the separating vertical expansion. It is possible to view an example in figure 5.43 on the facing page. This example shows the transition from an acceptable relation (a) to an unacceptable relation (b) back to an acceptable relation (c).

To summarize the discussion about vertical move of Interaction Occurrences with Interaction Occurrence relations, we can look briefly at the acceptable relation.

The first relation can lead to unacceptable relations A, B and C, but these can be fixed with the separating vertical expansion operation.

### 5.5.2 Horizontal move of Interaction Occurrence

**Messages**

There are two acceptable Interaction Occurrence/Message relations:

1. A Message outside an Interaction Occurrence and not connected to the Interaction Occurrence

2. A Message outside an Interaction Occurrence and connected to the Interaction Occurrence

And there are four unacceptable Interaction Occurrence/Message relations:

A. A Message inside an Interaction Occurrence and not connected to the Interaction Occurrence
Figure 5.43: Vertical move of Interaction Occurrence with Interaction Occurrence
B. A Message inside an Interaction Occurrence and connected to the Interaction Occurrence

C. A Message connected to and intersecting an Interaction Occurrence

D. A Message not connected to and intersecting an Interaction Occurrence

We will discuss horizontal Interaction Occurrence move in an acceptable diagram and see if it can lead to an unacceptable diagram.

The first relation is a Message being outside and not connected to an Interaction Occurrence. An example can be seen in (a) of figure 5.44 on the next page. This relation can end up in two different unacceptable relations (A and D) if the Interaction Occurrence is moved horizontally. These two unacceptable relations are the inside not connected to the Interaction Occurrence and intersecting the Interaction Occurrence. (b) of figure 5.44 on the facing page shows the intersecting relation. We will solve both unacceptable relations with the separating vertical expansion operation. After the expansion the Message will be outside of the Interaction Occurrence. See (c) as a result of the expansion performed on (b).

The second relation is a Message outside and connected with an Interaction Occurrence in the form of a Gate. View (a) of figure 5.45 on the next page for an example. After a horizontal Interaction Occurrence move we can get the two unacceptable relations B and C. Relation B is shown in (b) of figure 5.45 on the facing page. If we end up in one of these two unacceptable relations we will not be able to fix these with the expansion operations. Therefore we suggest to undo this move if it ends up in one of these two relations.

To summarize the discussion about horizontal move of Interaction Occurrences with Message relations, we can look briefly at the two acceptable relations.

The first relation can lead to unacceptable relations A and D, but it can be fixed with the separating vertical expansion operation.

The second relation can lead to unacceptable relations B and C, not fixable with expansion operations. We will therefore suggest to undo the edit operations resulting in this.

**Lifelines**

There are no unacceptable relations between Interaction Occurrences and Lifelines, therefore we will not discuss Interaction Occurrence/Lifeline relations.

**Combined Fragments**
Figure 5.44: Horizontal move of Interaction Occurrence with Message - 1st relation

Figure 5.45: Horizontal move of Interaction Occurrence with Message - 2nd relation
There are two acceptable Interaction Occurrence/Combined Fragment relations:

1. A Combined Fragment contains an Interaction Occurrence

2. A Combined Fragment and an Interaction Occurrence outside of each other

And there are two unacceptable relations:

A. A Combined Fragment inside an Interaction Occurrence

B. A Combined Fragment and an Interaction Occurrence intersects each other

We will see if the acceptable relations can be turned into unacceptable by a horizontal Interaction Occurrence move.

The first acceptable relation is an Interaction Occurrence inside a Combined Fragment. View (a) of figure 5.46 on the next page for an example of this relation. A horizontal move of an Interaction Occurrence can end up in the unacceptable relation B. See (b) of figure 5.46 on the facing page for an example this. We will suggest using separating vertical expansion to fix this. The result of performing this on (b) can be seen in (c) of figure 5.46 on the next page.

The second acceptable relation is an Interaction Occurrence and a Combined Fragment being outside of each other. Performing a horizontal Interaction Occurrence move can result in both unacceptable Interaction Occurrence/Combined Fragment relations. Both unacceptable relations can be solved in the same way as mentioned in the discussion of the first relation.

To summarize the discussion about horizontal move of Interaction Occurrences with Combined Fragment relations, we can look briefly at the two acceptable relations.

The first relation can lead to unacceptable relation B, but it can be fixed with the separating vertical expansion operation.

The second relation can lead to both unacceptable relations A and B, but these can be fixed with the separating vertical expansion operation.

**Interaction Occurrences**

There is one acceptable relation between Interaction Occurrences:

1. Two Interaction Occurrences outside of each other

And three unacceptable relations:

A. An Interaction Occurrence inside an Interaction Occurrence
Figure 5.46: Horizontal move of Interaction Occurrence with Combined Fragment - 1st relation
B. An Interaction Occurrence contains an Interaction Occurrence

C. Two Interaction Occurrences intersects each other

We are looking for situations where the acceptable relation becomes an unacceptable relation because of a horizontal Interaction Occurrence move.

The first and only acceptable relation is Interaction Occurrences outside of each other. View (a) of figure 5.47 for an example of this relation. The horizontal Interaction Occurrence move can result in all three unacceptable relations. In the example in (b) of figure 5.47 we see the move result in relation C. Using the separating vertical expansion placing the two Interaction Occurrences in one of the three unacceptable relations outside of each other will solve this. See (c) of figure 5.47 for the result of this on the unacceptable relation in (b).

To summarize the discussion about horizontal move of Interaction Occurrences with Combined Fragment relations, we can look briefly at the acceptable relation.

Figure 5.47: Horizontal move of Interaction Occurrence with Interaction Occurrence

(a) Before move

(b) After move

(c) After separating vertical expansion
The first relation can lead to unacceptable relations A, B and C, but these can be fixed with the separating vertical expansion operation.

### 5.5.3 Vertical scale of Interaction Occurrence

Vertical scale can be split into two, scale to make the Interaction Occurrence larger or scale to make the Interaction Occurrence smaller. We will call the two different scales for \textit{upsize-scale} and \textit{downsize-scale} respectively.

**Messages**

There are two acceptable Interaction Occurrence/Message relations:

1. A Message outside an Interaction Occurrence and not connected to the Interaction Occurrence
2. A Message outside an Interaction Occurrence and connected to the Interaction Occurrence

And there are four unacceptable Interaction Occurrence/Message relations:

A. A Message inside an Interaction Occurrence and not connected to the Interaction Occurrence
B. A Message inside an Interaction Occurrence and connected to the Interaction Occurrence
C. A Message connected to and intersecting an Interaction Occurrence
D. A Message not connected to and intersecting an Interaction Occurrence

We will discuss vertical Interaction Occurrence scale in an acceptable diagram and see if it can lead to an unacceptable diagram.

The first relation is a Message being outside and not connected to an Interaction Occurrence. An example can be seen in (a) of figure 5.48 on the next page. A vertical upsize-scale of an Interaction Occurrence in this relation can result in two of the four unacceptable Interaction Occurrence/Message relations. It is possible to end up in relations A and D. An example of relation A can be seen in (b) of figure 5.48 on the following page. This can be solved if we apply the separating vertical expansion on the diagram. In (c) of figure 5.48 on the next page this is done to fix (b). A vertical downsize-scale will not create any problems.

The second relation is a Message outside and connected with an Interaction Occurrence in the form of a Gate. View (a) of figure 5.49 on page 105
Figure 5.48: Vertical scale of Interaction Occurrence with Message - 1st relation
for an example. A vertical upsize-scale of an Interaction Occurrence in this relation will not create any problems. A vertical downsize-scale on the other hand can lead to a problem. As seen in (b) of figure 5.49. This means the Message is no longer connected to the Interaction Occurrence. It is neither connected to anything else in the end previously connected to the Interaction Occurrence. No amount of expansion, neither vertical nor horizontal, can fix this. We will therefore recommend to undo the scale.

To summarize the discussion about vertical scale of Interaction Occurrences with Message relations, we can look briefly at the two acceptable relations.

The first relation can lead to unacceptable relations A and D, but these can be fixed with the separating vertical expansion operation.

The second relation can end in an undefined unacceptable relation not fixable with expansion operations. We will therefore suggest to undo the edit operations resulting in this.

**Lifelines**

There are no unacceptable relations between Interaction Occurrences and Lifelines, therefore we will not discuss Interaction Occurrence/Lifeline relations.

**Combined Fragments**

There are two acceptable Interaction Occurrence/Combined Fragment relations:

1. A Combined Fragment contains an Interaction Occurrence
2. A Combined Fragment and an Interaction Occurrence outside of each other

And there are two unacceptable relations:

A. A Combined Fragment inside an Interaction Occurrence

B. A Combined Fragment and an Interaction Occurrence intersects each other

We will see if the acceptable relations can be turned into unacceptable by a vertical Interaction Occurrence scale.

The first acceptable relation is an Interaction Occurrence inside a Combined Fragment. View (a) of figure 5.50 on the facing page for an example of this relation. A vertical upsize-scale of an Interaction Occurrence in this relation can result in the unacceptable relation B. (b) of figure 5.50 on the next page shows this unwanted effect of a upsize-scale. We have had this exact same problem before and to solve it we used the separating vertical expansion operator. We suggest using it again. See figure (c) for the result of this procedure performed on the figure in (b). Downsize-scale will not cause any problems.

The second acceptable relation is an Interaction Occurrence and a Combined Fragment being outside of each other. This can result in both illegal relations A and B, but these errors can be solved in the same way as the previous.

To summarize the discussion about vertical scale of Interaction Occurrences with Combined Fragment relations, we can look briefly at the two acceptable relations.

The first relation can lead to unacceptable relation B, but it can be fixed with the separating vertical expansion operation.

The second relation can lead to unacceptable relations A and B, but these can be fixed with the separating vertical expansion operation.

**Interaction Occurrences**

There is one acceptable relation between Interaction Occurrences:

1. Two Interaction Occurrences outside of each other

And three unacceptable relations:

A. An Interaction Occurrence inside an Interaction Occurrence

B. An Interaction Occurrence contains an Interaction Occurrence
Figure 5.50: Vertical scale of Interaction Occurrence with Combined Fragment - 1st relation
C. Two Interaction Occurrences intersects each other

We are looking for situations where the acceptable relation becomes an unacceptable relation because of a vertical Interaction Occurrence scale.

The first and only acceptable relation is Interaction Occurrences outside of each other. It is possible to end up in all three unacceptable relations. Unacceptable relations A and B are always present at the same time. This leads to the idea that maybe there is no need to separate these two relations. But back to the discussion. It is possible to solve all three unacceptable relations using the separating vertical expansion operation. We will therefore recommend to perform this to fix these relations.

To summarize the discussion about vertical scale of Interaction Occurrences with Interaction Occurrence relations, we can look briefly at the acceptable relation.

The first relation can lead to unacceptable relations A, B and C, but these can be fixed with the separating vertical expansion operation.

5.5.4 Horizontal scale of Interaction Occurrence

Horizontal scale can be split into two, scale to make the Interaction Occurrence larger or to scale to make the Combined Fragment smaller. We will call the two different scales for *upscale-scale* and *downscale-scale* respectively.

Messages

There are two acceptable Interaction Occurrence/Message relations:

1. A Message outside an Interaction Occurrence and not connected to the Interaction Occurrence

2. A Message outside an Interaction Occurrence and connected to the Interaction Occurrence

And there are four unacceptable Interaction Occurrence/Message relations:

A. A Message inside an Interaction Occurrence and not connected to the Interaction Occurrence

B. A Message inside an Interaction Occurrence and connected to the Interaction Occurrence

C. A Message connected to and intersecting an Interaction Occurrence

D. A Message not connected to and intersecting an Interaction Occurrence
We will discuss horizontal Interaction Occurrence scale in an acceptable diagram and see if it can lead to an unacceptable diagram.

The first relation is a Message being outside and not connected to an Interaction Occurrence. This relation can lead to unacceptable relations A and D. An example can be seen in (a) of figure 5.51. The sequence seen in (a),(b) and (c) will be the typical cycle we suggest for a horizontal scale of an Interaction Occurrence resulting in unacceptable relation A. First the scale is performed and then the unacceptable relation is fixed by a separating vertical expansion.

The second relation is a Message outside and connected to an Interaction Occurrence in the form of a Gate. View (a) of figure 5.52 on the next page for an example. This relation can in fact result in unacceptable relations B and C. See (b) of figure 5.52 on the following page for an example of relation C. Expansion will not change these unacceptable relations to acceptable relations. We will therefore recommend using the undo operation.
Figure 5.52: Horizontal scale of Interaction Occurrence with Message - 2nd relation

To summarize the discussion about horizontal scale of Interaction Occurrences with Message relations, we can look briefly at the two acceptable relations.

The first relation can lead to unacceptable relations A and D, but it can be fixed with the separating vertical expansion operation.

The second relation can lead to unacceptable relations B and C. These are not fixable with expansion operations. We will therefore suggest to undo the edit operations resulting in this.

**Lifelines**
There are no unacceptable relations between Interaction Occurrences and Lifelines, therefore we will not discuss Interaction Occurrence/Lifeline relations.

**Combined Fragments**
There are two acceptable Interaction Occurrence/Combined Fragment relations:

1. A Combined Fragment contains an Interaction Occurrence
2. A Combined Fragment and an Interaction Occurrence outside of each other

And there are two unacceptable relations:

A. A Combined Fragment inside an Interaction Occurrence
Figure 5.53: Horizontal scale of Interaction Occurrence with Combined Fragment - 1st relation

We will see if the acceptable relations can be turned into unacceptable by a horizontal Interaction Occurrence scale.

The first relation is an Interaction Occurrence inside a Combined Fragment. An example can be seen in (a) of figure 5.53. This relation can turn into unacceptable relation B after a horizontal Interaction Occurrence. We recommend solving this error the same as every time we get intersecting Interaction Occurrences and Combined Fragments. We will suggest to use the separating vertical expansion operation. (a), (b) and (c) of figure 5.53 shows an example of this.
The second relation is an Interaction Occurrence and a Combined Fragment being outside of each other. This relation can turn into both unacceptable Interaction Occurrence/Combined Fragment relations. These can be solved with use of the separating vertical expansion operation.

To summarize the discussion about horizontal scale of Interaction Occurrences with Combined Fragment relations, we can look briefly at the two acceptable relations.

The first relation can lead to unacceptable relation B, but it can be fixed with the separating vertical expansion operation.

The second relation can lead to unacceptable relations A and B, but these can be fixed with the separating vertical expansion operation.

**Interaction Occurrences**

There is one acceptable relation between Interaction Occurrences:

1. Two Interaction Occurrences outside of each other

And three unacceptable relations:

A. An Interaction Occurrence inside an Interaction Occurrence

B. An Interaction Occurrence contains an Interaction Occurrence

C. Two Interaction Occurrences intersects each other

We are looking for situations where the acceptable relation becomes an unacceptable relation because of a horizontal Interaction Occurrence scale.

The first and only acceptable relation is Interaction Occurrences outside of each other. All three unacceptable relations can be the result of a horizontal Interaction Occurrence scale, but if the separating vertical expansion operation is used these three unacceptable relations will become the only acceptable relation. We will therefore recommend this.

To summarize the discussion about horizontal scale of Interaction Occurrences with Interaction Occurrence relations, we can look briefly at the acceptable relation.

The first relation can lead to unacceptable relations A, B and C, but these can be fixed with the separating vertical expansion operation.

### 5.5.5 Summary of Interaction Occurrence editing behavior

There are nine unacceptable relations between Interaction Occurrences and other elements. These are shown in table 5.10 on the next page.
<table>
<thead>
<tr>
<th>Element</th>
<th>Violating relation</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message</td>
<td>A Message inside an Interaction Occurrence and not connected to the Interaction Occurrence</td>
<td>MSG1</td>
</tr>
<tr>
<td></td>
<td>A Message inside an Interaction Occurrence and connected to the Interaction Occurrence</td>
<td>MSG2</td>
</tr>
<tr>
<td></td>
<td>A Message connected to and intersecting an Interaction Occurrence</td>
<td>MSG3</td>
</tr>
<tr>
<td></td>
<td>A Message not connected to and intersecting an Interaction Occurrence</td>
<td>MSG4</td>
</tr>
<tr>
<td>Combined Fragment</td>
<td>A Combined Fragment inside an Interaction Occurrence</td>
<td>CF1</td>
</tr>
<tr>
<td></td>
<td>A Combined Fragment and an Interaction intersects each other</td>
<td>CF2</td>
</tr>
<tr>
<td>Interaction Occurrence</td>
<td>An Interaction Occurrence inside an Interaction Occurrence</td>
<td>IO1</td>
</tr>
<tr>
<td></td>
<td>An Interaction Occurrence contains an Interaction Occurrence</td>
<td>IO2</td>
</tr>
<tr>
<td></td>
<td>Two Interaction Occurrences intersects each other</td>
<td>IO3</td>
</tr>
</tbody>
</table>

Table 5.10: Summary of unacceptable Interaction Occurrence relations
<table>
<thead>
<tr>
<th>Interaction Occurrence</th>
<th>Edit operation</th>
<th>Violating relation</th>
<th>Editor reaction</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Vertical move</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSG1</td>
<td>Separating vertical expansion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSG4</td>
<td>Separating vertical expansion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CF1</td>
<td>Separating vertical expansion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CF2</td>
<td>Separating vertical expansion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IO1</td>
<td>Separating vertical expansion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IO2</td>
<td>Separating vertical expansion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IO3</td>
<td>Separating vertical expansion</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Horizontal move</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSG1</td>
<td>Separating vertical expansion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSG2</td>
<td>Undo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSG3</td>
<td>Undo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSG4</td>
<td>Separating vertical expansion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CF1</td>
<td>Separating vertical expansion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CF2</td>
<td>Separating vertical expansion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IO1</td>
<td>Separating vertical expansion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IO2</td>
<td>Separating vertical expansion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IO3</td>
<td>Separating vertical expansion</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Vertical scale</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSG1</td>
<td>Separating vertical expansion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSG4</td>
<td>Separating vertical expansion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CF1</td>
<td>Separating vertical expansion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CF2</td>
<td>Separating vertical expansion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IO1</td>
<td>Separating vertical expansion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IO2</td>
<td>Separating vertical expansion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IO3</td>
<td>Separating vertical expansion</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Horizontal scale</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSG1</td>
<td>Separating vertical expansion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSG2</td>
<td>Undo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSG3</td>
<td>Undo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSG4</td>
<td>Separating vertical expansion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CF1</td>
<td>Separating vertical expansion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CF2</td>
<td>Separating vertical expansion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IO1</td>
<td>Separating vertical expansion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IO2</td>
<td>Separating vertical expansion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IO3</td>
<td>Separating vertical expansion</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.11: Summary of Interaction Occurrence editing behavior
In table 5.11 on the facing page we summarize the effect of the four edit operations discussed for Interaction Occurrences. The table shows which unacceptable relation the different editing operations can lead to. The table also summarizes how we recommended for the editor to react if the edit led to an unacceptable relation.

All nine unacceptable relations shows up in table 5.11. This means we have proposed an editor reaction to all nine relations. With use of the separating vertical expansion operation it is possible to resolve seven of the nine unacceptable relations. The two last relations, are not as easily resolved. The suggested editor reaction is to undo any edit operation leading to these unacceptable relations.

It was also found that the abstraction level of the unacceptable relations between Interaction Occurrences maybe was not as high as it could be. This could have been pursued if the analysis was put through another iteration.

5.6 Acceptable/unacceptable Lifeline relations

To help in the discussion about editing behavior of Lifelines we look at the possible relations a Lifeline can have with other elements. We will decide what is acceptable and what is unacceptable.

To differentiate if the relation is unacceptable because of the specification or our additional constraints we will create two unacceptable categories. If the relation is unacceptable according to the specification it will be a specification breaking relation. If it is unacceptable because of one of our additional constraints we will refer to it as a readability reducing relation.

5.6.1 Messages

There are three relations a Lifeline can have with a Message:

1. A Message is connected to a Lifeline
2. A Message intersects a Lifeline
3. A Message is neither connected to nor intersectes a Lifeline
4. A Message intersects the head of a Lifeline

A Lifeline can only be connected to Messages via the line of the Lifeline. We have also made the assumption that Messages can not appearing over the bottom of the Lifeline heads. This is done at the start of this chapter. We can therefore disregard the fourth relation.
The first relation is a Message connected to a Lifeline. If it is connected the Lifeline will not be intersected by that Message. It is possible to view figure 5.54 as an example. In this figure Lifeline c1 is connected to both Message msg1 and Message msg2. Lifeline c2 is connected to Message msg1 and Lifeline c3 is connected to Message msg2. This is acceptable according to the specification. The readability is not affected so we will allow this relation. Therefore the relation is acceptable.

The second relation is the intersect relation. When a Lifeline's line is intersected by a Message we have this relation. For an example see figure 5.54. Here Message msg2 intersects Lifeline c2. The specification allows this and so will we. The readability will not diminish greatly by allowing this. Therefore this relation is acceptable.

The third and last relation is a Message neither connected to nor intersecting a Lifeline. In figure 5.54 Message msg1 have this relation to Lifeline c3. There are no constraints in the specification disallowing this. And when it comes to the readability of the diagram it definitely does not lower it. Therefore this relation is acceptable.

As a conclusion it can be said that all three relations between Messages and Lifelines are acceptable. This can be seen in table 5.12 on the facing page.

### 5.6.2 Lifelines

There are two relations between Lifelines. These are:

1. Intersecting Lifelines

---

Figure 5.54: Lifeline in relation with Message
<table>
<thead>
<tr>
<th>Relation</th>
<th>Assessment</th>
<th>Rejection reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Message is connected to a Lifeline</td>
<td>Acceptable</td>
<td>-</td>
</tr>
<tr>
<td>A Message intersects a Lifeline</td>
<td>Acceptable</td>
<td>-</td>
</tr>
<tr>
<td>A Message is neither connected to nor intersects a Lifeline</td>
<td>Acceptable</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 5.12: Interaction Occurrence/Message relations

(a) Intersecting Lifelines          (b) Not intersecting Lifelines

Figure 5.55: Lifeline in relation with Lifeline
2. Not intersecting Lifelines

The first relation is two Lifelines intersecting. When we say intersecting Lifelines we mean two Lifelines intersecting heads. Because we have the assumption that all Lifeline heads are aligned this relations is in fact two Lifelines sharing some common horizontal space in the diagram. See (a) of figure 5.55 on the page before for an example of this. There are no constraints handling this relation in the specification, but due to the reduced readability we will suggest that this is a readability reducing relation.

The second relation is not intersecting Lifelines. This can be seen in (b) of figure 5.55 on the preceding page. There are no constraints in the specification disallowing this. And when it comes to readability there are no problems. Therefore we will allow it. This relation is acceptable.

The result of the discussion is one acceptable relation and one unacceptable. See table 5.13 for the complete result.

### 5.6.3 Combined Fragments

Combined Fragment/Lifeline relations are discussed in section 5.2.2 on page 40. There are two acceptable relations. The result was summarized in table 5.2 on page 41.

### 5.6.4 Interaction Occurrences

Interaction Occurrence/Lifeline relations are discussed in section 5.4.2 on page 86. There are two acceptable relations. Table 5.8 on page 87 showed the result of this discussion.
5.7 Lifeline editing behavior

5.7.1 Vertical move of Lifeline

Because of the assumption that all Lifeline heads are aligned vertical move of Lifelines can not be performed. Therefore we will not discuss this edit operation.

5.7.2 Horizontal move of Lifeline

Messages
There are no unacceptable relations between Lifelines and Messages, therefore we will not discuss Lifeline/Message relations.

Lifelines
There is one acceptable Lifeline/Lifeline relation:

1. Not intersecting Lifelines

And one unacceptable relation:

A. Intersecting Lifelines

We will find out if a vertical move of a Lifeline with the acceptable relation can end up in the unacceptable relation.

The first relation is two Lifelines not intersected. This relation can be seen in (a) of figure 5.56 on the following page. A horizontal move of one of the Lifelines can lead to intersecting Lifelines. See (b) of the same figure. This is an unacceptable relation an must be fixed. One suggestion is to perform horizontal expansion between the two intersecting Lifelines. This expansion should only have effect on one of the Lifelines. This will be the horizontal equivalence to the separating vertical expansion. We will therefore call it separating horizontal expansion. This will turn the unacceptable relation into a acceptable relation. (c) shows the result of this done to (b).

To summarize the discussion about horizontal move of Lifelines with Lifeline relations, we can look briefly at the acceptable relation.

The first relation can lead to unacceptable relation A, but it can be fixed with the separating horizontal expansion operation.

Combined Fragments
There are no unacceptable relations between Lifelines and Combined Fragments, therefore we will not discuss Lifeline/Combined Fragment relations.
(a) Lifelines not intersecting

(b) Horizontal move of c2 leads to intersection

(c) Separating horizontal expansion between c1 and c2 restores the acceptable relation

Figure 5.56: Horizontal move of Lifeline in relation with Lifeline - 1st relation
Interaction Occurrences
There are no unacceptable relations between Lifelines and Interaction Occurrences, therefore we will not discuss Lifeline/Interaction Occurrence relations.

5.7.3 Vertical scale of Lifeline
Vertical scale can be split into two, scale to make the Lifeline larger or to scale to make the Lifeline smaller. We will call the two different scales for *upsise-scale* and *downsize-scale* respectively. Because of the assumption that all Lifeline heads are aligned we will not allow scaling at the top of a Lifeline.

Messages
There are no unacceptable relations between Lifelines and Messages, therefore we will not discuss Lifeline/Message relations.

Lifelines
There is one acceptable Lifeline/Lifeline relation:

1. Not intersecting Lifelines

And one unacceptable relation:

A. Intersecting Lifelines

We will find out if a vertical scale of a Lifeline with the acceptable relation to another Lifeline can end up in the unacceptable Lifeline/Lifeline relation.

The first relation is two Lifelines not intersected. And because of the restriction made about aligned Lifeline heads it is only possible to perform upsize-scale and downsize-scale of the bottom of the Lifeline. This will not lead to intersecting Lifelines. An example can be seen in figure 5.57 on the next page.

To summarize the discussion about horizontal move of Lifelines with Lifeline relations, we can look briefly at the acceptable relation.

The first relation creates no problems.

Combined Fragments
There are no unacceptable relations between Lifelines and Combined Fragments, therefore we will not discuss Lifeline/Combined Fragment relations.

Interaction Occurrences
There are no unacceptable relations between Lifelines and Interaction Occurrences, therefore we will not discuss Lifeline/Interaction Occurrence relations.
5.7.4 Horizontal scale of Lifeline

Vertical scale can be split into two, scale to make the Lifeline larger or to scale to make the Lifeline smaller. We will call the two different scales for *upsise-scale* and *downsize-scale* respectively.

**Messages**

There are no unacceptable relations between Lifelines and Messages, therefore we will not discuss Lifeline/Message relations.

**Lifelines**

There is one acceptable Lifeline/Lifeline relation:

1. Not intersecting Lifelines

And one unacceptable relation:

A. Intersecting Lifelines

We will find out if a horizontal scale of a Lifeline with the acceptable relation can end up in the unacceptable relation.

The first relation is two Lifelines not intersected. Horizontal scale of one of the Lifelines can lead to intersecting Lifelines. This can be seen in figure 5.58 on page 124. The figure shows also that by performing a separating horizontal expansion between the Lifelines this unacceptable relation can be turned into an acceptable relation. We will therefore recommend using this expansion on this unacceptable relation.
Table 5.14: Summary of Lifeline editing behavior

<table>
<thead>
<tr>
<th>Edit operation</th>
<th>Violating relation</th>
<th>Editor reaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horizontal move</td>
<td>Intersecting Lifelines</td>
<td>Separating horizontal expansion</td>
</tr>
<tr>
<td>Vertical scale</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Horizontal scale</td>
<td>Intersecting Lifelines</td>
<td>Separating horizontal expansion</td>
</tr>
</tbody>
</table>

To summarize the discussion about horizontal move of Lifelines with Lifeline relations, we can look briefly at the acceptable relation.

The first relation can lead to unacceptable relation A, but it can be fixed with the separating horizontal expansion operation.

**Combined Fragments**
There are no unacceptable relations between Lifelines and Combined Fragments, therefore we will not discuss Lifeline/Combined Fragment relations.

**Interaction Occurrences**
There are no unacceptable relations between Lifelines and Interaction Occurrences, therefore we will not discuss Lifeline/Interaction Occurrence relations.

### 5.7.5 Summary of Lifeline editing behavior

In table 5.14 we summarize the effect of the three edit operations discussed for Lifelines. The table shows which unacceptable relation the different editing operations can lead to. The table also summarizes how we recommended for the editor to react if the edit led to an unacceptable relation.

Vertical scale does not lead to any unacceptable relations. But both horizontal edit operations can lead to the only unacceptable relation for Lifelines. This is suggested resolved with the separating horizontal expansion operation.

### 5.8 Acceptable/unacceptable Message relations

To help in the discussion about editing behavior of Messages we look at the possible relations a Message can have with other elements. We will decide what is acceptable and what is unacceptable.

To differentiate if the relation is unacceptable because of the specification or our additional constraints we will create two unacceptable categories. If
(a) Lifelines not intersecting

(b) Horizontal scale of c1 leads to intersection

(c) Separating horizontal expansion between c1 and c2 restores the acceptable relation

Figure 5.58: Horizontal scale of Lifeline in relation with Lifeline - 1st relation
the relation is unacceptable according to the specification it will be a specification breaking relation. If it is unacceptable because of one of our additional constraints we will refer to it as a readability reducing relation.

5.8.1 Messages

Messages have three possible relations to other Messages:

1. Two Messages intersecting
2. Two Messages not intersecting
3. Overlapping Messages

The first relation is two Messages intersecting each other. See (a) of figure 5.59 on the following page for an example. We will also include the case where a Message End is intersected by another Message in this relation. See (b) of figure 5.59. The specification does not have a constraint disallowing this relation. The readability can be diminished, but the meaning of this relation is rather clear. Therefore we will allow this. The relation is acceptable.

The second relation is two Messages not intersecting. By this we mean Messages not in contact with each other. See (c) of figure 5.59 on the next page for an example. This is both acceptable according to the specification and easy to read. The relation is acceptable.

The third relation is overlapping Messages. By overlapping we mean Messages covering each other like in (d) of figure 5.59 on the following page. In this figure there are two Messages, Message msg1 goes from Lifeline c1 to Lifeline c3 while Message msg2 goes from Lifeline c2 to Lifeline c4. The line between Lifeline c2 and c3 is the area of the overlap. There are no constraints handling this relation in the specification. But the readability is obviously reduced. It is for example not possible to determine if Message msg2 starts from Lifeline c1, Lifeline c2 or Lifeline c3. Therefore we will suggest this is a readability reducing relation.

The discussion is summarized in table 5.15 on page 127.

5.8.2 Lifelines

Lifeline/Message relations are discussed in section 5.6.1 on page 115. Out of three relations all three relations are acceptable. See table 5.12 on page 117 for a summary of the discussion.
Figure 5.59: Message in relation with Message
5.8.3 Combined Fragments

Combined Fragment/Message relations are discussed in section 5.2.1 on page 36. Out of five relations four are acceptable and one is unacceptable. The result can be seen in table 5.1 on page 39.

5.8.4 Interaction Occurrences

Interaction Occurrence/Message relations are discussed in section 5.4.1 on page 83. Out of five relations two are acceptable and three are unacceptable. The result can be seen in table 5.7 on page 86.

5.9 Message editing behavior

In the previous sections we have seen how to solve different unacceptable relations between Combined Fragments, Interaction Occurrences, Lifelines and Messages. We have seen that the same unacceptable relations can be dealt with in the same way. We will therefore not look at Message editing behavior with regards to other elements. But we will discuss Messages in relations with other Messages.

When it comes to possible editing operations of Messages it is possible to think of two different move operations. It would be moving either one Message End or two Message Ends. Moving of one Message End can be seen in figure 5.60 on the following page This means a Message can for example go from being connected to a Lifeline and become connected to an Interaction Occurrence instead. We will call this Message End move.

Moving two Message Ends at the same time can be implemented as a vertical move. That is, the relative position between the two Message Ends is the same before and after the move, and the Message will be connected to the same elements. This can be seen in figure 5.61 on the next page. We will call this vertical move of a Message.

<table>
<thead>
<tr>
<th>Relation</th>
<th>Assessment</th>
<th>Rejection reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two Messages intersecting</td>
<td>Acceptable</td>
<td>-</td>
</tr>
<tr>
<td>Two Messages not intersecting</td>
<td>Acceptable</td>
<td>-</td>
</tr>
<tr>
<td>Two Messages overlapping</td>
<td>Unacceptable</td>
<td>Readability reducing</td>
</tr>
</tbody>
</table>

Table 5.15: Interaction Occurrence/Message relations
Figure 5.60: Message End move

Figure 5.61: Vertical move of Message
5.9.1 Message End move

There are two acceptable relations between Messages.

1. Intersected

2. Not intersected

We will again use an acceptable diagram as the basis of the discussion. And we will see if a move of one of the Message Ends can result in this unacceptable relation:

A. Overlapping Messages

The first relation is two Messages intersecting. If we have the special case of intersect, where a Message End is intersected by another Message, a move of a Message End can result in the unacceptable relation A. This can point in the direction that the intersect relation is at a too high abstraction level. In figure 5.62 it is possible to see an example of this. We can use the separating vertical expansion operation. This would mean pushing one of the Messages down-wards, while the other is unaffected by the expansion.

The second relation is not intersecting Messages. This relation can also end up in relation A after a move of a Message End. This can be seen in figure 5.63 on the following page. And to fix this unacceptable relation we can use the same procedure as shown in the first relation.

To summarize the discussion about horizontal move of Lifelines with Lifeline relations, we can look briefly at the acceptable relations.
The first relation can lead to unacceptable relation A, but it can be fixed with the separating vertical expansion operation.

The second relation can lead to unacceptable relation A, but it can be fixed with the separating vertical expansion operation.

5.9.2 Vertical move of Message

There are two acceptable relations between Messages.

1. Intersected

2. Not intersected

We will again use an acceptable diagram as the basis for the discussion. And we will see if a vertical move of a Message can result in the unacceptable relation with another Message. The unacceptable relation is:

A. Overlapping Messages

The first relation is the intersect relation. For two Messages to intersect they can not be parallel, and for two Messages to overlap they need to be parallel. Vertical move of a Message will not alter the skewness of the Message. Two intersected Messages will therefore never become overlapping Messages as a result of a vertical move of one of the Messages.
The second relation is the not intersecting relation. This relation can result in relation A. See figure 5.64 for an example of this. Making use of the separating vertical expansion operation will again solve the conflict.

To summarize the discussion about horizontal move of Lifelines with Lifeline relations, we can look briefly at the acceptable relations.

The first relation creates no problems.

The second relation can lead to unacceptable relation A, but it can be fixed with the separating vertical expansion operation.

### 5.9.3 Summary of Message editing behavior

In table 5.16 we summarize the effect of the two edit operations discussed for Messages. The table shows which unacceptable relation the different editing operations can lead to. The table also summarizes how we recommended for the editor to react if the edit led to an unacceptable relation. The one unacceptable Message relation can be resolve with the use of the separating vertical expansion operation.

It was also found that the intersect relation maybe should be divided into intersection of two Message lines and intersection of a Message End.

<table>
<thead>
<tr>
<th>Message</th>
<th>Violating relation</th>
<th>Editor reaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message End move</td>
<td>Overlapping Messages</td>
<td>Separating vertical expansion</td>
</tr>
<tr>
<td>Vertical move</td>
<td>Overlapping Messages</td>
<td>Separating vertical expansion</td>
</tr>
</tbody>
</table>

Table 5.16: Summary of Message editing behavior

Figure 5.64: Vertical move of Message in relation with Message
Chapter 6

Conclusions and further work

6.1 Conclusions

6.1.1 SeDi

The SeDi editor is an almost complete UML 2.0 sequence diagram editor with syntactical support. It has implemented most of the constructs found in the specification, it can generate files used for diagram interchange and can be integrated with other tools.

6.1.2 Sequence diagram editing analysis

We have seen a systematic way of analyzing the editing operation of a sequence diagram editor.

The idea was to first define the acceptable and unacceptable relations between different elements in a sequence diagram. Then the different editing operations for each element was discussed. The goal of this is to find which acceptable relations can lead to unacceptable relations. When this was done it was possible to look at how these unacceptable relations can be turned into acceptable relation.

The analysis showed that the same unacceptable relations could be resolved in the same way. This was independent of which edit operation was performed and which acceptable relation was the basis for the discussion.

A challenge with the analysis is to divide the unacceptable relations sufficiently so it is possible to resolve one relation in one way. One way of doing this is to perform the analysis in an iterative way. This would mean to start with a rather high level of abstraction for the relations. After one iteration the abstraction level can be adjusted.
6.2 Further work

6.2.1 SeDi

For further work on the SeDi editor it is possible to try implementing the results from the analysis. This would increase the usability of the SeDi editor.

Implementing the results of the analysis would mean two things. It would mean to implement a mechanism for checking if there are unacceptable relations in a diagram. And it would mean implementing the different editor reactions recommended in the discussion.

Because the internal model of SeDi is rather limited it would be necessary to create the associations needed for checking for unacceptable relations. It would be possible to use the same techniques used when creating the UML2 model. This is already implemented and this parses the SeDi model in a way sufficient for checking for unacceptable relations. This can be done every time an element is edited.

In the event of an unacceptable relation the editor should then react in the same way as recommended in the discussion. This can for example be the undo operation. There is already implemented an undo operation in SeDi. The next step could therefore be to let SeDi use the undo operation as a reaction to an edit operation resulting in an unacceptable relation.

Would also be beneficial to make SeDi capable of taking UML2 files as input. One way of doing this is to try to do the opposite of the UML2 model generation described in section 3.2.

It could also be an idea to try to use the UML2 file as the only model repository for SeDi. That is instead of having a special SeDi model (described in section 3.1) it has to be created a UML2 file from, the only model is a UML2 model.

It could also be interesting to see if it could be possible to make SeDi expandable in the form of the user adding new elements. This would lead, for example, to incorporate a UML 2.0 testing profile into SeDi. For more about UML 2.0 testing profiles see [2].

It would also be possible to try to incorporate SeDi closer with IBM Rational Software Modeler. This could lead to a fully compatible plug-in to IBM Rational Software Modeler.
Bibliography


Appendix A

SeDi

A.1 SeDi UML2 generation code

This is an excerpt of the code that generates the UML2 file in the SeDi editor. The complete code is bundled with the SeDi distribution.

```java
package seDi.uml2;

import java.io.IOException;

import seDi.model.*;

public class GenerateUML2 extends DebugUML2 {

    public int generateUML2(SeDiEditor editor) {

        registerResourceFactories();

        IFile file = ((IFileEditorInput)editor.getEditorInput()).getFile();
        String name = file.getName();

        model = createModel(name.substring(0, name.length() - 3));
        collaboration = createCollaboration(model);

        /*
         * Accessing the frame in the diagram
         *
         */
        Frame frame;
```
List children=editor.getDiagram().getAllChildren();
if(children.size()==1&&(children.get(0) instanceof Frame)){
    frame=(Frame)children.get(0);
} else {
    System.out.println("Not possible to create uml2 file");
    return -1;
}

interaction = createInteraction(collaboration,frame);
otherElements = findCombinedFragments(findLifelines(frame));
createLifelines();

/* Adding messages to the otherElements list */
findMessages(frame);
for(int i=0;i<listOfMessages.size();i++){
    otherElements.add(listOfMessages.get(i));
}

/* Adding combined fragments to the otherElements list and removing elements inside combined fragment from the otherElements list */
findCFElements();
for(int i=0;i<listOfCF.size();i++){
    otherElements.add(listOfCF.get(i));
}

/* sorting the otherElements list so creation will be in the right order (from the top left and down) */
otherElements=sortList(otherElements,true);
createElements(otherElements);
fixRef();
if(!valid){
    err("Could not create uml2-model");
    return -1;
    //TODO - get the dialog working
}

/* Finding the location and name of the source .sd file */
* Creating a *.uml2 file in the same location and with the same name *

```java
if (file.getFileExtension().toString().equals("sd")){
    String filePath = file.getLocation().toOSString();
    filePath = filePath.substring(0, filePath.length() - 3);
    URI paths = URI.createFileURI(filePath);
    paths = paths.appendFileExtension(UML2Resource.FILE_EXTENSION);
    save(model, paths);
}
else{
    System.out.println("Error! Wrong fileformat on sourcefile. Must be .sd file.");
    return -1;
}
return 1;
}

private Model createModel(String name) {
    Model model = UML2Factory.eINSTANCE.createModel();
    model.setName(name);
    return model;
}

private Collaboration createCollaboration(Model model) {
    Collaboration c = (Collaboration) model.createOwnedMember(UML2Package.eINSTANCE.getCollabc);
    c.setName("Null");
    return c;
}

private Property createProperty(Collaboration collaboration, Class curr, seDi.model.Lifeline currLifeli) {
    Property p = (Property) collaboration.createOwnedAttribute(UML2Package.eINSTANCE.getProper;
    int index = currLifeline.getText().indexOf(":");
    if (index > -1){
        p.setName(currLifeline.getText().substring(0, index));
    } else
    p.setName(curr.getName());
    return p;
}
```
private Interaction createInteraction(Collaboration c, Frame frame) {
    Interaction i = (Interaction)c.createOwnedBehavior(UML2Package.eINSTANCE.getInteraction());
    i.setName(frame.getText());
    return i;
}

private void createElements(List elementList) {
    SeDiSubpart tmp;
    while(elementList.size()>0){
        tmp=(SeDiSubpart)elementList.get(0);
        if(tmp instanceof SeDi.model.Message){
            createMessage((SeDi.model.Message)tmp,interaction);
        }
        else if(tmp instanceof IO){
            createIO((IO)tmp,interaction);
        }
        else if(tmp instanceof CF){
            createCF((CF)tmp,interaction);
        }
        else if(tmp instanceof Continuation){
            createContinuation((Continuation)tmp,interaction);
        }
        else if(tmp instanceof Terminate){
            createTermination((Terminate)tmp,interaction);
        }
        else if(tmp instanceof Coregion){
            createCoregion((Coregion)tmp,interaction);
        }
        else if(tmp instanceof EO){
            createEO((EO)tmp,interaction);
        }
        elementList.remove(tmp);
    }
}

private void createEO(EO eo, Interaction interA) {
    ExecutionOccurrence uml2eo = (ExecutionOccurrence)interA.createFragment(UML2Package.eINSTANCE.getExecutionOccurrence());
    List coveredLifelines = findCoveredLifelines(eo);
    List cover=uml2eo.getCovereds();
    if(coveredLifelines.size()<1) {
        return;
    }
    for(int i=0;i<coveredLifelines.size();i++){
        cover.add(uml2Elements.get(coveredLifelines.get(i)));
    }
    EventOccurrence e1 = (EventOccurrence) interA.createFragment(UML2Package.eINSTANCE.getEventOccurrence());
    e1.setStart(uml2eo);
    cover=e1.getCovereds();
    cover.add(uml2Elements.get(coveredLifelines.get(0)));
}
EventOccurrence e2 = (EventOccurrence) interA.createFragment(UML2Package.eINSTANCE.uml2eo.setFinish(e2);
cover=e2.getCovereds();
cover.add(uml2Elements.get(coveredLifelines.get(0)));

uml2Elements.put(eo,uml2eo);
}

private void createCoregion(Coregion coregion, Interaction interA)
{
    // TODO - this is shorthand for a parallel combined fragment
    // each of the elements inside the coregion is an operand...
}

private void createTermination(Terminate terminate, Interaction interA)
{
    Stop stop = (Stop)interA.createFragment(UML2Package.eINSTANCE.getStop());
    List coveredLifelines = findCoveredLifelines(terminate);
    List cover=stop.getCovereds();
    for(int i=0;i<coveredLifelines.size();i++){
        cover.add(uml2Elements.get(coveredLifelines.get(i)));
    }
    uml2Elements.put(terminate,stop);
}

private void createContinuation(Continuation continuation, Interaction interA)
{
    org.eclipse.uml2.Continuation c = (org.eclipse.uml2.Continuation)interA.createFragment(UM
    List coveredLifelines = findCoveredLifelines(continuation);
    List cover=c.getCovereds();
    for(int i=0;i<coveredLifelines.size();i++){
        cover.add(uml2Elements.get(coveredLifelines.get(i)));
    }
    c.setName(continuation.getText());
    uml2Elements.put(continuation,c);
}

private void createCF(CF cf, Interaction interA)
{
    CombinedFragment uml2cf = (CombinedFragment)interA.createFragment(UML2Package.eIN
    List coveredLifelines = findCoveredLifelines(cf);
    List covereds=uml2cf.getCovereds();
    for(int i=0;i<coveredLifelines.size();i++){
        covereds.add(uml2Elements.get(coveredLifelines.get(i)));
    }
    setRightOperator(uml2cf,cf.getText());
    createOperands(uml2cf, cf, coveredLifelines);
private void createOperands(CombinedFragment uml2cf, CF cf, List coveredLifelines) {
    List sep = findOperands(cf);
    InteractionOperand operand;

    List children = sortList(cf.getAllChildren(), true);
    Rectangle rect;
    for(int j = 0; j < sep.size(); j++) {
        operand = (InteractionOperand) uml2cf.createOperand(UML2Package.eINSTANCE.getInteractionOp);
        rect = getCFRectangle(cf, j, sep);
        createCFElements(operand, cf, rect, children);
    }
    cf.getAllChildren().clear();
}

private void createCFElements(InteractionOperand operand, CF cf, Rectangle r, List children) {
    SeDiSubpart part;
    for(int i = 0; i < children.size(); i++) {
        part = (SeDiSubpart) children.get(i);
        if(insideOperand(part, r)) {
            if(part instanceof seDi.model.Message) {
                createMessage((seDi.model.Message) part, operand);
            } else if(part instanceof IO) {
                createIO((IO) part, operand);
            } else if(part instanceof CF) {
                createCF((CF) part, operand);
            } else if(part instanceof EO) {
                createEO((EO) part, operand);
            } else if(part instanceof Terminate) {
            }
        }
    }
}
```java
private void createIO(IO io, Interaction interA) {
    InteractionOccurrence interO = (InteractionOccurrence) interA.createFragment(UML2Package.eINSTANCE);
    List coverLifelines = findCoveredLifelines(io);
    List covereds = interO.getCovereds();
    for (int i = 0; i < coverLifelines.size(); i++) {
        covereds.add(uml2Elements.get(coverLifelines.get(i)));
    }
    interO.setName(io.getText());
    uml2Elements.put(io, interO);
}

private void createLifelines() {
    scDi.model.Lifeline currLifeline;
    Class currClass;
    Property currProperty;
    org.eclipse.uml2.Lifeline currUML2Lifeline;
    for (int i = 0; i < listOfLifelines.size(); i++) {
        currLifeline = (scDi.model.Lifeline) listOfLifelines.get(i);
        currClass = createClass(model, currLifeline);
        classes.add(currClass);
        currProperty = createProperty(collaboration, currClass, currLifeline);
        properties.add(currProperty);
        currUML2Lifeline = createLifeline(interaction, currProperty);
        uml2Elements.put(currLifeline, currUML2Lifeline);
    }
}

private org.eclipse.uml2.Lifeline createLifeline(Interaction interaction, Property currProperty) {
    org.eclipse.uml2.Lifeline l = (org.eclipse.uml2.Lifeline) interaction.createLifeline(UML2Package.eINSTANCE);
    l.setName(currProperty.getName());
    l.setRepresents(currProperty);
    return l;
}

private Model createModel(String name) {
    Model model = UML2Factory.eINSTANCE.createModel();
```
model.setName(name);

return model;
}

private Collaboration createCollaboration(Model model) {
    Collaboration c = (Collaboration) model.createOwnedMember(UML2Package.eINSTANCE.getCollaboration());
    c.setName("null");

    return c;
}

private Property createProperty(Collaboration collaboration, Class curr, seDi.model.Lifeline currLifeline) {
    Property p = (Property) collaboration.createOwnedAttribute(UML2Package.eINSTANCE.getProperty());
    p.setType((Type) curr);
    int index = currLifeline.getText().indexOf(" :");
    if (index >= 1) {
        p.setName(currLifeline.getText().substring(0, index));
    } else p.setName(curr.getName());
    return p;
}

private Interaction createInteraction(Collaboration c, Frame frame) {
    Interaction i = (Interaction) c.createOwnedBehavior(UML2Package.eINSTANCE.getInteraction());
    i.setName(frame.getText());
    return i;
}

private org.eclipse.uml2.Class createClass(Model model, seDi.model.Lifeline l) {
    org.eclipse.uml2.Class c = (Class) model.createOwnedMember(UML2Package.eINSTANCE.getClass());
    int index = l.getText().indexOf(" :");
    if (index >= 1 && l.getText().length() > 1) {
        c.setName(l.getText().substring(index + 1));
    } else c.setName(l.getText());
    return c;
}

/**
 * Nesting CF
 */
private void addCFToCF() {
    List tmpList = new ArrayList();
    Object tmp;

    /*
    * Nesting CF
    */
}
for(int i=0;i<listOfCF.size();i++){  
    tmp=listOfCF.get(i);  
    for(int j=0;j<listOfCF.size();j++){  
        if(j!=i&&inside((CF)tmp,(CF)listOfCF.get(j))){  
            ((CF)listOfCF.get(j)).add((CF)tmp);  
            tmpList.add(tmp);  
            break;  
        }  
    }  
}  
for(int i=0;i<tmpList.size();i++){  
    listOfCF.remove(tmpList.get(i));  
}  

private void findCFElements() {  
    listOfCF=sortList(listOfCF,true);  
    Object tmp;  
    /*  
     * Sorting the cf list so the addOtherElementsToCF works  
     */  
    for(int j=0;j<listOfCF.size();j++)  
        for(int i=j;i>0 & inside((CF)listOfCF.get(i),(CF)listOfCF.get(i-1));i--){  
            tmp=listOfCF.get(i);  
            listOfCF.set(i,listOfCF.get(i-1));  
            listOfCF.set(i-1,tmp);  
        }  
    addOtherElementsToCF();  
    addCFToCF();  
}

private List findCoveredLifelines(SeDiSubpart part) {  
    List coveredBy = new ArrayList();  
    seDi.model.Lifeline tmp;  
    int xcoord,partxcoord;  
    for(int i=0;i<listOfLifelines.size();i++)  
        tmp = (seDi.model.Lifeline)listOfLifelines.get(i);  
        xcoord = tmp.getLocation().x+tmp.getSize().width/2;  
        partxcoord = part.getLocation().x;  
        if(xcoord>partxcoord & xcoord<(part.getSize().width+partxcoord))  
            coveredBy.add(tmp);  
    return coveredBy;  
}
private List findLifelines(Frame frame) {
    List elements = new ArrayList();
    List children = frame.getAllChildren();

    SeDiSubpart tmp;
    for (int i = 0; i < children.size(); i++) {
        tmp = (SeDiSubpart) children.get(i);
        if (tmp instanceof seDi.model.Lifeline) {
            listOfLifelines.add(tmp);
        } else {
            elements.add(tmp);
        }
    }
    listOfLifelines = sortList(listOfLifelines, false);
    return elements;
}

private List sortList(List unsortedList, boolean ycoord) {
    List sortedList = new ArrayList();
    seDi.model.SeDiSubpart tmp;
    while (unsortedList.size() > 0) {
        tmp = removeMin(unsortedList, ycoord);
        sortedList.add(tmp);
    }
    return sortedList;
}
A.2 SeDi user manual

This is the user manual distributed with the SeDi editor.
About SeDi 1.1.0

This tool (SeDi) has been made by Andreas Limyr as part of his Master’s work. For the sake of the thesis which is under production it would be very interesting to receive comments and information about how this work has been applied in your institutions.

This version of SeDi (1.1.0) has little support for checking syntax constraints. Therefore it is possible to create diagrams that are not legal UML 2 sequence diagrams. What comes out of trying to generate a UML2 repository from such illegal diagrams is unknown and the production is not guaranteed to succeed. However production of UML2 repository from correct UML 2.0 sequence diagrams should result in a correct repository. Even illegal diagrams will normally produce a repository of unknown quality.

The SeDi tool only concentrates on sequence diagrams. Therefore the necessary elements of the context are simply inferred from the sequence diagram.

For known faults, problems and limitations see the Known Issues file.

Andreas Limyr
Department of Informatics
University of Oslo
Norway
March 2005
Adding figures

In general the symbol menu items should be clicked and the corresponding element placed on the canvas.

**Frame**: The only symbol allowed in an empty sd-file, and only one frame is allowed in one file. Click the left button to place upper left corner of a default frame that can then be scaled to desired shape.

**Lifeline**: Click to place lifeline horizontally. All lifelines are placed horizontally aligned. Click again on the selected lifeline and the text can be edited. In general pressing F2 when there is a selected item will also make it possible to edit its text. Lifelines can be moved horizontally and stretched downwards by scaling.

**CF**: Combined Fragment: Click to place upper left corner of a default combined fragment frame. Scale to desired size. Edit the operator name.

**Separator**: Click to place one end of the separator. Then click to place other end. Placing the second end can only be done when the separator is horizontal. A separator cannot be modified (in this version) so changes to its placement must be done by deleting it and inserting a new one.

**IO**: Interaction Occurrences: Click to place upper left corner of a default interaction occurrence. Scale to correct size. Edit text.
In this version combined fragments and interaction occurrences are not glued to the lifelines, but are contained within the enclosing interaction frame.

**Coregion**: Click to place upon a lifeline. In this version it does not snap onto the lifeline. The coregions are not considered when producing UML2 repositories.

**EO**: Execution Occurrences: Click to place upon lifeline. In this version it does not snap onto the lifeline.

**Termination**: Click to place upon lifeline. In this version termination symbol does not snap onto the lifeline.

**Continuation**: Click to place a default continuation that can be scaled to desired size. The continuation does not glue to the covered lifelines, but the correct UML2 repository is produced nevertheless.

**Message**: Click to place sending end of the message first. Then click to place the receiving end. The cursor symbol will indicate with a "no parking" sign when the placement is illegal. Message ends may appear on lifelines (where they snap) and to interaction occurrences, combined fragments and interactions (frame) as gates. Sometimes it is significant on which side of a line you place the message end. During the editing of the receiving end the message line will be displayed. The placed message will have a default text. The message creation button is “sticky” that is you will continue to create messages until you choose another button on the symbol menu. To select a message you need to click "selection" menu item and then the message. Then the text can be edited.

A message can be modified by repositioning either message ends. The cursor will change shape to indicate when it is within reach of a message end.
Creation of a new sd-file

If you do not have a project to put your sd-file into you must create one.
To create a new project, go to File -> New -> Project.
Choose Simple->Project. Give it a name and you have created a simple project.

Now for creating a sd-file.
Go to File -> New -> Example. Locate and select the icon named SD and push Next.
Fill in the required information and push Finish. Be sure name the file with a sd extension.

Congratulations you have created your first sd-file!
Delete

To delete a figure select it and click the right mouse button to access the delete command. Delete is also possible with a group of figures. Messages connected to deleted figures will also be deleted.
**Group**

Select an area by dragging the cursor. All non-message items within the area will be selected. Shift drag is also possible to increase the selected set. Before moving the selected set click on the Selection button.
Known issues

1. **Messages**
   It is only possible to move a message by moving the message ends separately.
   Messages will not group together with other elements when grouping.

2. **Interaction Occurrences**
   All Lifelines will be created to be covered by all Interaction Occurrences. It is therefore not possible to have a Lifeline

3. **Separator**
   When creating a Separator you must see to it that it is straight.
   Moving a Separator is not possible. You must therefore delete the one you want to move and create another one in the new location.

4. **Editing**
   When performing an undo after a delete of a fragment connected to messages the messages will not reappear.
   There are no copy and paste implementations.

5. **UML2 files**
   At this time the tool only supports production of UML2 files and not creation of *.sd files from UML2 files.
   Generated Interaction Occurrences does not have a reference to the Interaction referred to in the “Reference here” field.
   Messages crossing Combined Fragments will not be generated correctly. If a Message is sent from a Lifeline outside the Combined Fragment to a Lifeline inside the Combined Fragment it will be handled like the Message is not inside the Combined Fragment at all. (And vice versa)
   Default Combined Fragment operator is “seq”.

   Class and Property generation is done based on Lifeline information. The pattern used is <property>:<class>
   - Coregion is not implemented. This means that a Coregion symbol in a diagram will not result in anything in the UML2 file.

6. **Print action**
   Only tested for Windows.

If you have an opinion about this program please send an e-mail

This tool (SeDi) has been made by Andreas Limyr as part of

Andreas Limyr
Department of Informatics
University of Oslo
Norway
March 2005
**Printing diagram**

It is possible to print the diagram (Only tested for Windows).
Select the .sd file in the Navigator window.
Press the right mouse button and then select Print.
You will be prompted with a Select Print Mode dialog.
You get the following options:

- **Tile** is the default print mode. It prints diagram at 100% scale and tiles horizontally and/or vertically, if necessary.
- **Fit page** scales the diagram so that the entire printed image fits on one page.
- **Fit width** scales the diagram so that the width of the printed image fits on one page and tiles vertically, if necessary.
- **Fit height** scales the diagram so that the height of the printed image fits on one page and tiles horizontally, if necessary.

Select your wanted mode and press **OK**.
Renaming figures

Select the figure you want to rename. Press the right mouse button and select Rename. Or you can select the figure and then reselect it again. Or you can select the figure and push the F2 button. Write in the new text. When you are finished press enter or select another figure.
**Saving**

See to it that the editor is the active window.
Save your work by push `ctrl-S`.
Clicking the right mouse button and selecting **Save** will also save your sd-file.
You can see if you have unsaved edits by looking for a * in front of the filename in the editor tab.
It there is a * you have unsaved edit operations.
Selection

Click to select. Shift-click will make multiple select. As will control-click. Control-click gives the possibility to unselect figures in the multiple select.
Creating a project from SeDi the source

If you want to look at the source code or even try to develop SeDi you can import the source code into a plug-in project. Start with File -> Import. Select External Plug-ins and Fragments and click Next. The Import From field should be marked to The target platform. (Unless you have installed SeDi somewhere else than the Eclipse installation) Under Plug-ins and Fragments to Import select the Select from all plug-ins…. option. And finally under Import As select Projects with source folders. Push Next. Locate seDi in the list over Plug-ins and Fragments Found. Mark seDi (not seDi.help!) and push the Add --> to choose seDi for import. Push Finish. You will now have a plug-in project with the source code of SeDi.

To run SeDi goto Run -> Run As -> Run-time Workbench. The SeDi used in the Run-Time Workbench is built from the plug-in project.
Undo

If you want to undo an action press the right mouse button and select **Undo**. This will undo the latest action independent of your selection.