

Design and implementation of a software module for multilocus sequence typing

Diploma Thesis

Agnes Angelika Thanbichler

December 19, 2007



Prof. Dr. Bernhard Haubold
FH Weihenstephan
Fakultät Biotechnologie und Bioinformatik
85350 Freising
Germany



Dr. Roald Forsberg
CLC bio A/S
Gustav Wieds Vej 10
8000 Aarhus C
Denmark

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass die vorliegende Arbeit von mir selbst und ohne fremde Hilfe verfasst und noch nicht anderweitig für Prüfungszwecke vorgelegt wurde. Es wurden keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt. Wörtliche und sinngemäße Zitate sind als solche gekennzeichnet.

Freising, den

.....
Agnes Thanbichler

Acknowledgements

First of all, I want to thank my supervisor at CLC bio, Dr. Roald Forsberg, for his supporting advices and excellent suggestions during the thesis. I also want to thank Prof. Dr. Bernhard Haubold, my supervisor at the university, for being such a great help. Keith A. Jolley (PubMlst [1]) provided lots of information and help, I am very thankful for this. Many thanks go also to the brilliant team at CLC bio, it was fun and pleasure working with them all. Last but not least, I am deeply grateful to Stefan M. Schuster for numerous fruitful discussions and his motivations.

Contents

1	Summary	1
2	Introduction	2
2.1	Form and content of the thesis	2
2.2	Background	2
2.3	Typing	3
2.3.1	Phenotypic method	4
2.3.2	Genotypic method	4
2.4	CLC bio	4
3	Multilocus sequence typing	6
3.1	Introduction	6
3.2	Singlelocus sequence typing	8
3.3	Comparison to other typing methods	8
3.3.1	Multilocus enzyme electrophoresis (MLEE)	8
3.3.2	Pulsed-field gel electrophoresis (PFGE)	8
3.4	MLST scheme	9
3.4.1	Sequences	9
3.4.2	Profiles	11
3.5	Public databases	11
3.6	PubMlst	11
3.6.1	Profiles databases	11
3.6.2	Isolates databases	14
3.7	Workflow	14
3.7.1	Preparation	14
3.7.2	Multilocus sequence typing	14
3.7.3	Multilocus sequence analysis	17
4	Development	18
4.1	Motivation of the work	18
4.2	Aim of the work	19
4.3	Existing solutions	19
4.3.1	Online form	19
4.3.2	Open source software / Freeware	20
4.3.3	Applied Biosystems	20
4.4	End-user and product analysis	20
4.5	Software development process	21
4.5.1	Feature Driven Development (FDD)	21
4.5.2	Test Driven Development (TDD)	22
4.5.3	FDD-TDD approach	22
4.6	Java	24
4.7	JUnit	24

4.8	CLC bio's Software Developer Kit	24
5	Requirements specification	25
5.1	Purpose	25
5.2	Scope	25
5.3	Product perspective	25
5.4	Product features	25
5.5	User classes and characteristics	27
5.5.1	Overview	27
5.5.2	Detailed model	27
5.6	System features	28
5.6.1	Features list	29
5.6.2	Typing workflow	29
5.6.3	Create MLST scheme	30
5.6.4	Download MLST scheme	32
5.6.5	Merge MLST schemes	33
5.6.6	Extend MLST scheme	35
5.6.7	Extend Isolate	36
5.6.8	Submission of data	38
5.7	Non-functional requirements	38
5.7.1	Performance requirements	38
5.7.2	Software Quality Attributes	39
6	Realization/Implementation	39
6.1	Persistence	39
6.2	System features	40
6.2.1	Typing workflow	40
6.2.2	Create MLST scheme	48
6.2.3	Download MLST scheme	48
6.2.4	Merge MLST scheme	50
6.2.5	Extend MLST scheme	51
6.2.6	Extend Isolate	52
6.2.7	Submission of data	53
6.3	Additional features	53
6.3.1	Automatic update	53
6.3.2	Popup menus in views	54
6.3.3	History	54
6.3.4	Undo/Redo	55
6.4	Graphical user interface	55
6.4.1	History view	55
6.4.2	Isolate editors and views	55
6.4.3	MLST scheme editors and views	59
7	Results	62

7.1 Overall evaluation	62
7.2 Detailed evaluation	63
7.3 User acceptance	64
8 Discussion	65
9 Conclusion	67
10 References	68
Appendices	72
A PubMlst download - dbase.xml	73
B Glossary & Acronyms	75
C List of Figures	79
D List of Tables	82

1 Summary

Multilocus sequence typing (MLST) proposed by Maiden et al. [2] had become a standard in typing of microorganisms in the mid-nineties. *MLST* means that an *isolate*, the pure culture of an organism, can be distinguished from other *isolates* by comparing its *Deoxyribonucleic acid (DNA)* sequence fragments of housekeeping genes. A *housekeeping gene* encodes proteins, necessary for the basic maintenance, core functions and metabolism of a cell. It is constitutively expressed, that means the expression of the gene is not regulated, which turns *housekeeping genes* into good candidates for *MLST*.

An advantage of *MLST* is the opportunity of building databases with known typed sequences. Some public databases, like PubMlst.org [1], have a web-based form to type bacterial sequences. But there are redundant work steps and the web-based form is not very flexible, e. g. the typing of sequences is not persisted and users are dependent on different curators. Curators are responsible for new entries in databases, which is an advantage regarding accuracy, however, it takes much time.

The aim of this work is to create a fast and userfriendly workflow for typing bacterial sequences, which has additional functionality to facilitate the work regarding *multilocus sequence typing*.

The result of this work is a software module, the *CLC MLST Module*. The data of the databases hosted on PubMlst.org [1] is accessible by download functionality of the *CLC MLST Module*. All data acquired during *multilocus sequence typing* can be extended and changed locally. The workflow and the dynamic typing of the sequences was implemented. Furthermore additional functionality was designed and implemented in order to ease the handling of *MLST* data.

2 Introduction

2.1 Form and content of the thesis

This section gives information about the form and content of this thesis and therefore assists the reading.

Form

All italicized terms can be found either in the Glossary or the Acronyms (Appendix B). If you search a glossary entry or an acronym in the text, the formatted terms will help you to find it very fast. An example is “*multilocus sequence typing*”. Additionally, the page numbers of the appearances are shown in the Glossary and Acronyms (Appendix B).

Content

The following pages of Section 2 (Introduction) familiarize the principal topics of this work. The Section 3 (Multilocus sequence typing) describes *multilocus sequence typing* and introduces public databases and the workflow of *MLST*. Afterwards all design issues and utilities used for the implementation of the work are stated in “Development” in Section 4. Next two sections show the requirements specification and implementation (Section 5, Section 6). Finally results, discussion and the conclusion are elaborated (Section 7, Section 8, Section 9).

2.2 Background

Infectious diseases are a well known problem causing high morbidity and mortality rates as well as high costs. Molecular typing of *isolates* (the pure microbial sample) was introduced concerning tasks like:

- Identification of the source organism
- Distinction of infectious from noninfectious strains
- Distinction of relapse from reinfection

Molecular typing and the epidemiological surveillance have been proven to be very cost-effective by reason of the reduction of infections. The Northwestern Memorial Hospital in Chicago, Illinois, established an in-house molecular typing program [3].

“The rate of infection fell to 43 percent below the national average, and approximately 50 deaths were avoided during the 5-year period”, cited from [4].

2.3 Typing

The term *typing* stands for two main aspects, the classification of an organism and the strain identification. The latter aspect identifies variants of an organism. To classify an organism, it implies that an prior classification took place, like the Bergey's Manual [5]. In this thesis the term typing is used with respect to strain identification, the distinction between *isolates*. Attributes of a typing system are

- **Reproducibility**
The reproducibility refers to a method obtaining same results with every iteration performed of someone else. A lack of reproducibility could come from variation in the method and/or alteration in the microbial sample.
- **Specificity**
The specificity is the probability that a test of a binary classification (positive/negative) is negative if the sample is negative. As an example we use the test: "Is a microorganism a new type?". The specificity is then the probability of the case that the test returns "not new type" if the microorganism is not a new type. Specificity is the probability of true negative located test results.
- **Sensitivity**
Analogical to the specificity is the sensitivity which is the probability that a test is positive if the sample is positive. With the example: "Is a microorganism a new type?", the sensitivity is the probability of a test result "microorganism is new type" if the microorganism is a new type. Thus, the sensitivity is the probability of true positive located test results.
- **Discrimination**
A good typing system should distinguish between epidemiologically unrelated isolates, which ideally are assigned to different types.
- **Standardization**
The typing system should be standardized, that means that there is a unified and approved procedure to type in different laboratories with no lack of reproducibility. Often standardizations induce the opportunity to build databases and therefore a collaboration of different laboratories can be achieved.
- **Performance**
Performance is a main attribute of a typing system. Only if the system can deliver results fast it will be accepted. A good performance saves time and costs.
- **Effectiveness**
Investment reduction of time and money is an essential challenge therefore the gain of information with time and cost, the effectiveness, is of high impact.
- **Ease of typing**
The typing should be performed in an easy way that everyone can be trained

fast. The evaluation of the results has to be as easy as possible, since complex systems are hardly accepted.

There are two different ways to perform typing of microorganisms, the *phenotypic* and *genotypic* methods.

2.3.1 Phenotypic method

A *phenotypic* method carries out the typing with observable characteristics of an organism, like biotypes, serotypes, bacteriophage or bacteriocin types and antimicrobial susceptibility profiles. *Isolates*, which cannot be typed, are more common with *phenotypic* methods but can also occur in *genotypic* methods.

The main disadvantages of *phenotypic* methods are low reproducibility, as well as specific methods for specific organisms, which cannot be transferred to other organisms and thus lead to problematic standardization. *Phenotypic* methods are generally lower discriminatory than *genotypic* methods (Section 2.3.2).

2.3.2 Genotypic method

Typing based on molecular level (*DNA* and *Ribonucleic acid (RNA)*) is called *genotypic* method.

Main disadvantages of *genotypic* methods are the isolation of intact chromosomal *DNA* and sometimes *phenotypic* nature, like pathogenicity, cannot be determined. The detection of pathogenicity or *antibiotic* characteristics of an organism play an important role in molecular typing. *Genotypic* typing systems are especially suited for evolutionary studies. The benefit of the *genotypic* method is that all systems are based on the same information, the combination of four nucleotide bases (cytosine; guanine; adenine; thymine (*DNA*) or uracil (*RNA*)).

2.4 CLC bio

The company CLC bio [6] provides solutions in the area of bioinformatics (Figure 1).

The aim of this project was to build a plug-in with the Software Developer Kit of CLC bio, described in Section 4.8. Plug-ins are divided into two groups depending on their size and complexity:

Extensions Relatively small plug-ins with one or few features

Modules Large plug-ins that provide multiple features

The result of this work is the CLC MLST Module ([7]).

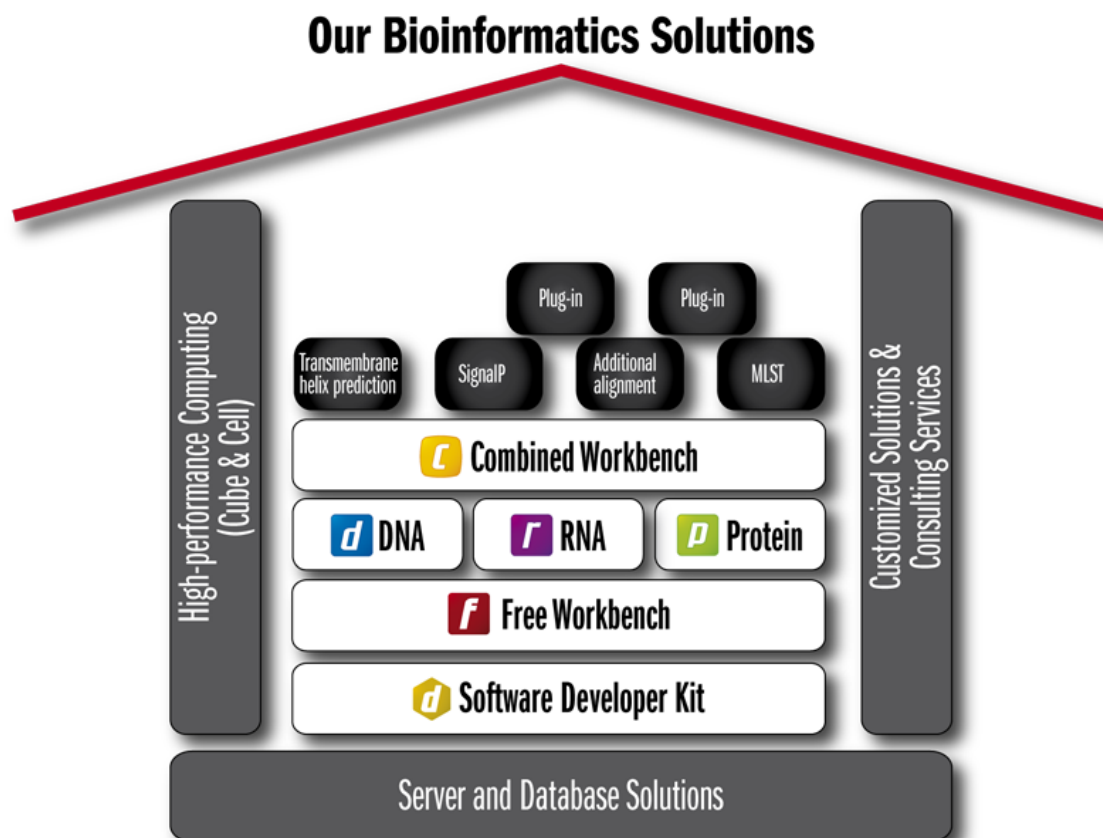


Figure 1: CLC bio's solutions

Main segments of CLC bio's [6] offers are software, consulting and high performance computing. The software segment is made up of solutions like the Combined (yellow), Protein (green), DNA (blue) and RNA (purple) Workbench, as well as the Software Developer Kit (golden) and the Plug-ins (black), as at November 2007.

3 Multilocus sequence typing

3.1 Introduction

Multilocus sequence typing is a *genotype*-based typing method. Commonly 1 to 11 *loci* (regions of the *genome* of an organism) of a chromosomal *DNA* sequence are used to determine a *sequence type*, the result of the *multilocus sequence typing*.

An *isolate* is typed by comparing *allele* sequence fragments of certain lengths of predefined *loci* to other *allele* sequences. Sequences, that differ in at least one nucleotide base, are seen as different *allele types*. All distinct *allele types* have unique numbers. The consequence of this comparison is a combination of numbers representing the *allele* sequences. Each of the number is representing an *allele type*, a specific *allele* sequence.

The composition of those *allele types* forms the *sequence type*, which also gets a unique number. *Sequence types* of different compositions of *allele types* have different numbers. The *sequence type* represents a specific combination of *allele type*.

The numbering was introduced in order to provide a fast and easy readability. Thus, it is quite easy to distinguish between two different *sequence types*, since they have different numbers. However, the numbering doesn't provide any information regarding relatedness between the sequences or *sequence types*.

All *sequence types* and the associated information about the sequences are collected in a so-called *MLST scheme*. *MLST schemes* are explained in Section 3.4.

MLST schemes are available for several bacterial organisms and fungal *MLST schemes* are now emerging ([8]).

Work related to *MLST* is splitted up into three steps, see also in Figure 2,

1. Preparation/Data collection

All work regarding determination of the nucleotide sequence, used for *MLST*, is done in this step. The isolation of the *DNA* from an organism and the *Polymerase chain reaction* are primary tasks.

2. *Multilocus sequence typing*

In general the assigning of an *isolate* to an existing or possible new *sequence type* is the main function of this step. Also maintenance of databases, for example adding new sequences and *sequence types*, is handled here.

3. *Multilocus sequence analysis*

All analyses, performed with respect to the data, which is created in step 1 and step 2, are seen as *Multilocus sequence analysis*.

To gain insight to the work of *MLST*, the Section 3.7 states the workflow of *MLST* in deeper matter.

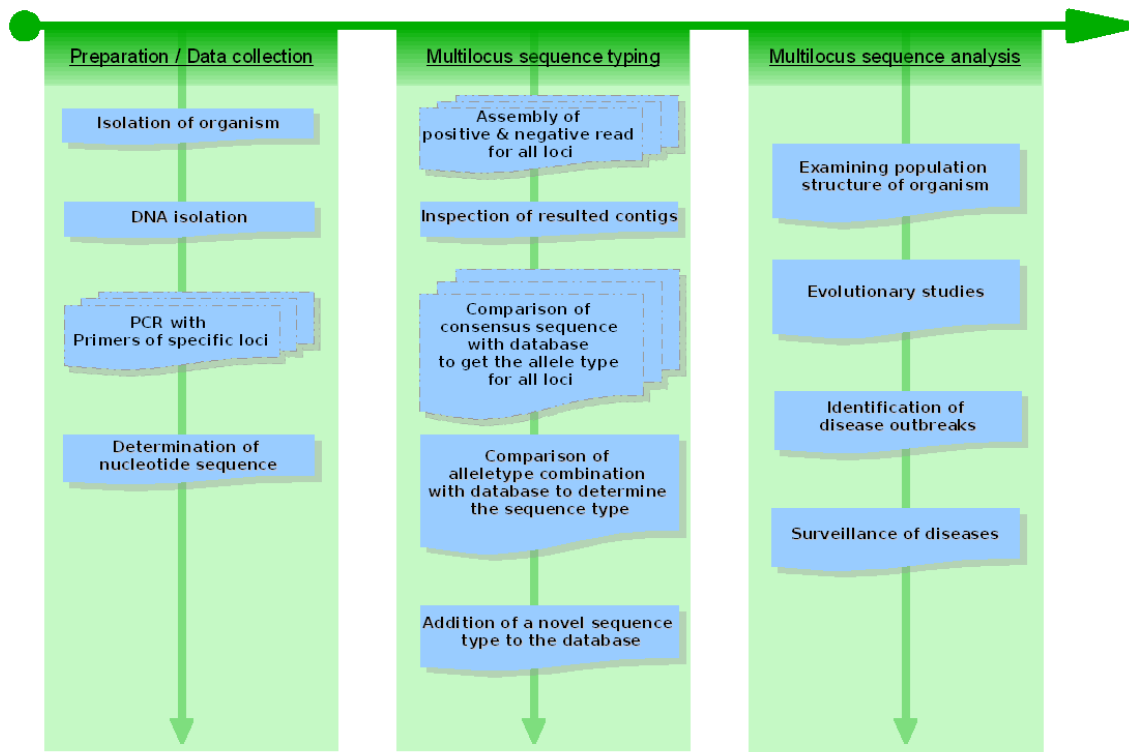


Figure 2: Overview of work regarding *MLST*. The result of the preparation step and *multilocus sequence typing* will be added to a database and is stored for more analyses, the *Multilocus sequence analyses*.

3.2 Singlelocus sequence typing

Singlelocus sequence typing is similar to *MLST*, however, it is used to type only with one *locus*, whereas *MLST* uses up to 11 *loci*.

Spa-typing is a well-established practice to type the organism *Staphylococcus aureus*, especially the *Methicillin-resistant Staphylococcus aureus (MRSA)*. This organism has a gene called “spa”, which is the *locus* for *singlelocus sequence typing*. Even though, there is a *MLST scheme* for the organism *Staphylococcus aureus* with more *loci*, *Spa-typing* had turned out to be an approved typing method [9, 10, 11].

3.3 Comparison to other typing methods

There are several applied typing systems which are mainly differentiated by the *phenotypic* and *genotypic* approach. Since sequencing costs and time decrease drastically the last years, the *genotypic* based methods gain more and more in importance. In this section the most commonly used methods are introduced and compared with *MLST*. The methods are contrasted in a table at the end of this section, see Table 1.

3.3.1 Multilocus enzyme electrophoresis (MLEE)

Multilocus enzyme electrophoresis (MLEE) is a *phenotypic* based method. Some encoded proteins of an organism are used to detect enzyme polymorphisms on basis of differing electrophoretic mobilities of those proteins on a gel.

MLEE is the precursor of *MLST* but with the major difference that *MLST* is based on the nucleotide information (*genotypic* approach). *MLEE* uses complete genes, whereas *MLST* uses fragments of genes. However, *MLEE* can only detect modifications in the gene, if it causes an alteration in the electrophoretic mobility of the protein.

3.3.2 Pulsed-field gel electrophoresis (PFGE)

Pulsed-field gel electrophoresis (PFGE) is a *genotypic* approach. Chromosomal *DNA* is digested with rare-cutting restriction enzymes, resulting in several fragments of different lengths. Those fragments are analysed by separation with agarose gel electrophoresis. Differences in the patterns of the separated fragments are referred to as restriction fragment length polymorphisms (RFLPs). The resulting *DNA*-fragments are too large to be separated by conventional agarose gel electrophoresis, which fail to separate fragments with more than 40 to 50 *kilo base pairs (kbp)* length. In *PFGE*, fragments of over 1000 *kbp* are separated by periodically changing the direction of the electrical field.

In general *PFGE* and *MLST* show similar levels of discrimination but sometimes *MLST* reveals a higher power of discrimination [12, 13, 14, 15, 16, 17]. However,

the limiting factors are the restriction enzymes in *PFGE* and the gene *loci* used in *MLST*. *PFGE* can detect large-scale genomic rearrangements which *MLST* is not able to identify therefore *PFGE* operates more discriminatory in species with mobile elements. Another main advantage of *PFGE* over *MLST* is, that it is carried out with the absence of any detailed information about the *genome*.

	<i>MLST</i>	<i>MLEE</i>	<i>PFGE</i>
TECHNIQUE	<i>Genotypic</i>	<i>Phenotypic</i>	<i>Genotypic</i>
REPRODUCIBILITY	High	Low	Moderate
DISCRIMINATION	High	Moderate - High	Moderate - High
STANDARDIZATION	High	Moderate	Moderate
PERFORMANCE	Low	Moderate	Moderate
EFFECTIVENESS	Moderate	Moderate	Moderate
EASE OF TYPING	High	Moderate	Moderate
IS CHROMOSOMAL INFORMATION NEEDED?	Yes	Yes	No

Table 1: Most commonly used typing techniques in molecular biology. Attributes, explained in Section 2.3, are compared of *MLST*, *MLEE* and *PFGE*.

3.4 MLST scheme

An *MLST scheme* consists of *allele* sequences (Section 3.4.1) and *profiles* (Section 3.4.2). *MLST schemes* define the genes and the fragment lengths in order to accurately type against it. It is not permitted to have redundant entries in an *MLST scheme*. The genes and their sequence lengths of an *MLST scheme* and the organism itself build the discriminatory power for the distinction of *isolates*. For instance, the *Mycobacterium tuberculosis* varies in 1/10000 nucleotides and would therefore hardly achieve a satisfactory discriminatory power ([18]).

3.4.1 Sequences

Figure 3a shows some of the sequences included in an *MLST scheme*. Usually the sequences of an *MLST scheme* represent only a fragment of the whole gene sequence of an organism. The length is around 400 - 600 *base pair (bp)*. This fragment length was chosen due to the reliably read of a single run of the gel-based automated sequencing instruments of the mid 1990s.

Sequences in an *MLST scheme* have an arbitrarily number. If two sequences differ in at least one nucleotide, they have to have different numbers, the *allele types*. In Figure 3a the *allele type* is contained in the name of each sequence. For example

```

>arcc-1
TTATTAATCCAACAAGCTAAATCGAACAGTGACACAACGCCGGCAATGCCATTGGATACT
TGTGGTGCAATGTGCACAGGGTATGATAGGCTATTGGTTGGAAACTGAAATCAATCGCATT
TAACTGAAATGAATAGTGATAGAACTGTAGGCACAATCGTTACACGTGTGGAAGTAGAT
AAAGATGATCCACGATTCAATAACCCAAACCAAAACCAATTGGTCCTTTTTATACGAAAGAA
GAAGTTGAAGAATTACAAAAAGAACAGCCAGACTCAGTCTTTAAAGAAAGATGCAGGACGT
GGTTATAGAAAAAGTAGTTGCGTCACCACTACCTCAATCTATACTAGAACACCAGTTAATT
CGAACTTTAGCAGACGGTAAAAATATTGTCATTGCATGCGGTGGTGGCGGTATTCCAGTT
ATAAAAAAGAAAAATACCTATGAAGGTGTTGAAGCG

>arcc-2
TTATTAATCCAACAAGCTAAATCGAACAGTGACACAACGCCGGCAATGCCATTGGATACT
TGTGGTGCAATGTGCACAGGGTATGATAGGCTATTGGTTGGAAACTGAAATCAATCGCATT
TAACTGAAATGAATAGTGATAGAACTGTAGGCACAATCGTAAACACGTGTGGAAGTAGAT
AAAGATGATCCACGATTTGATAACCCAACTAAACCAATTGGTCCTTTTTATACGAAAGAA
GAAGTTGAAGAATTACAAAAAGAACAGCCAGGCTCAGTCTTTAAAGAAAGATGCAGGACGT
GGTTATAGAAAAAGTAGTTGCGTCACCACTACCTCAATCTATACTAGAACACCAGTTAATT
CGAACTTTAGCAGACGGTAAAAATATTGTCATTGCATGCGGTGGTGGCGGTATTCCAGTT
ATAAAAAAGAAAAATACCTATGAAGGTGTTGAAGCG

>arcc-3
TTATTAATCCAACAAGCTAAATCGAACAGTGACACAACGCCGGCAATGCCATTGGATACT
TGTGGTGCAATGTGCACAGGGTATGATAGGCTATTGGTTGGAAACTGAAATCAATCGCATT
TAACTGAAATGAATAGTGATAGAACTGTAGGCACAATCGTTACACGTGTGGAAGTAGAT
AAAGATGATCCACGATTTGATAACCCAACTAAACCAATTGGTCCTTTTTATACGAAAGAA
GAAGTTGAAGAATTACAAAAAGAACAGCCAGACTCAGTCTTTAAAGAAAGATGCAGGACGT
GGTTATAGAAAAAGTAGTTGCGTCACCACTACCTCAATCTATACTAGAACACCAGTTAATT
CGAACTTTAGCAGACGGTAAAAATATTGTCATTGCATGCGGTGGTGGCGGTATTCCAGTT
ATAAAAAAGAAAAATACCTATGAAGGTGTTGAAGCG

```

(a) Some sequences of the gene “arcc” of organism *Staphylococcus aureus* from Mlst.net [19] in Fasta format.

ST	arcc	aroe	glpf	gmkn_	pta_	tpi_	yqil
1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	26
3	1	1	1	9	1	1	12
4	10	10	8	6	10	3	2
5	1	4	1	4	12	1	10
6	12	4	1	4	12	1	3
7	5	4	1	4	4	6	3
8	3	3	1	1	4	4	3
9	3	3	1	1	1	1	10
10	8	7	6	2	9	9	7

(b) Some profiles of the organism *Staphylococcus aureus* from Mlst.net [19].

Figure 3: Example of some data of a *MLST scheme*, the *Staphylococcus aureus* from Mlst.net [19].

“arcc-1” means, it is an allele sequence of gene *arcc* and has the *allele type* 1. This *allele type* is used to form the number combination for the *sequence type*, thus, the numbers in the *profiles*, except the numbers for the *sequence type*, are allele numbers.

3.4.2 Profiles

In Figure 3b, you can see some of the profiles of the *MLST scheme* of organism *Staphylococcus aureus* from Mlst.net [19]. Every line except the header represents a profile specific to a *sequence type*. The header contains the abbreviation for *sequence type* (ST), the names of the genes (*arcc*, *aroe*, *glpf*, *gmkl*, *pta*, *tpi*, *yqil*) and sometimes information about already performed analyses regarding clonal complexes (*clonal_complex*).

The profile gives information about the *alleles* of each gene for one *sequence type*. Figure 4 shows the composition of a *Sequence type (ST)* 8. All *isolates* with same profile must have same *sequence type*.

3.5 Public databases

There are several public available databases. The most known databases are listed in Table 2 with organisms they provide information for. Lots of other organisms are available as well as variable specifications for same organisms.

The databases of the different organisms listed in Table 2 are curated by particular elected persons, who have expert knowledge of the organism. Only valid data will be added and therefore those databases appear to be highly accurate.

3.6 PubMlst

The organisms hosted on PubMlst [1] are shown in Table 2. The database of a specific organism is splitted up into two sections, the Profiles database and the Isolates database.

3.6.1 Profiles databases

In this database, the *MLST scheme* information, the *profiles* and *allele* sequences, are hosted. Following forms are available in this database:

- Querying allele sequences
- Querying profiles
- Querying *sequence types*
- Searching the database
- Downloading profiles, alleles and *trace* files

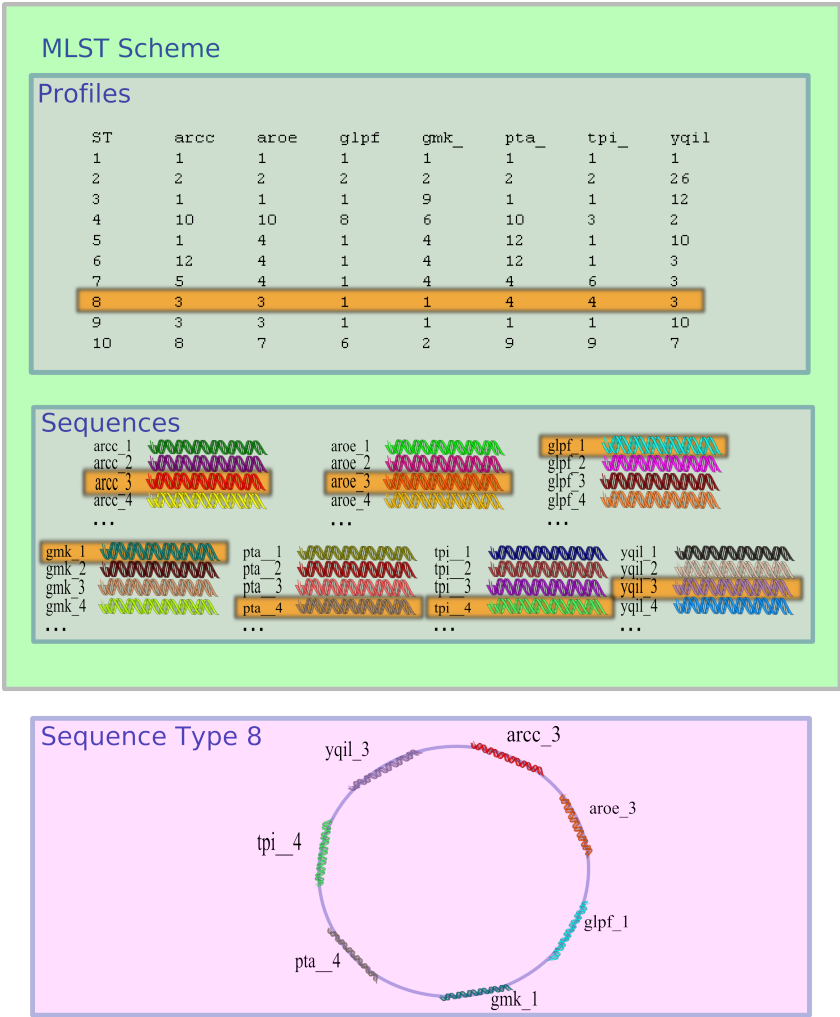


Figure 4: The connection between the *profile* and sequences is shown in this figure. Every number in the *profile* represents a sequence in the same *MLST scheme*. The combination of the numbers makes up a *sequence type*. All isolates of same *sequence type* must have the same sequence in every *locus*. The *sequence type* box contains a circular illustration of a bacterial chromosome, which comprises the sequences of an organism of *ST 8*.

DATABASE	ORGANISMS
PubMlst [1]	<p>Acinetobacter baumannii, Aspergillus fumigatus, Bacillus cereus, Bordetella spp., Burkholderia cepacia complex, Campylobacter fetus, Campylobacter helveticus, Campylobacter insulaenigrae, Campylobacter jejuni, Campylobacter coli, Campylobacter lari, Campylobacter upsaliensis, Helicobacter pylori, Klebsiella pneumoniae, Listeria monocytogenes, Neisseria spp., Porphyromonas gingivalis, Pseudomonas aeruginosa, Streptococcus agalactiae, Streptococcus uberis, Streptomyces spp., Vibrio parahaemolyticus, Vibrio vulnificus, Wolbachia spp., Candida krusei, Candida tropicalis</p>
Mlst.net [19]	<p>Burkholderia pseudomallei Candida albicans Candida glabrata Cryptococcus neoformans var grubii Enterococcus faecalis Enterococcus faecium Haemophilus influenzae Staphylococcus aureus Staphylococcus epidermidis Streptococcus pneumoniae Streptococcus pyogenes Streptococcus suis</p>
Max Planck Institute for Infection Biology [20]	<p>Escherichia coli Moraxella catarrhalis Salmonella enterica Yersinia pseudotuberculosis</p>
SpaServer [11]	Staphylococcus aureus

Table 2: Public available databases with organisms as at end of October 2007.

- Determining information about the polymorphic sites found at a locus - locus explorer

The Profile databases are used to request the *sequence type* of an *isolate*. The online input forms and example results for the organism *Streptococcus uberis* ¹ are demonstrated in Figure 5.

3.6.2 Isolates databases

All *isolates*, available for the organism, are hosted in the Isolates database. Additional information, like the source of the *isolate* (blood, skin, ...) and geographical information, is stored here. Parts of this section of the organisms database do not have public access. The Isolates database contains available forms:

- Querying by *sequence types*
- Querying by profile
- Searching the database
- Query by cited publication
- Data analysis
- Exporting data

3.7 Workflow

This section examines the standard workflow of *multilocus sequence typing*. Figure 2 shows an overall view of the tasks in a *MLST* workspace. Generally there are three steps

1. Preparation/Data collection
2. *Multilocus sequence typing*
3. *Multilocus sequence analysis*

3.7.1 Preparation

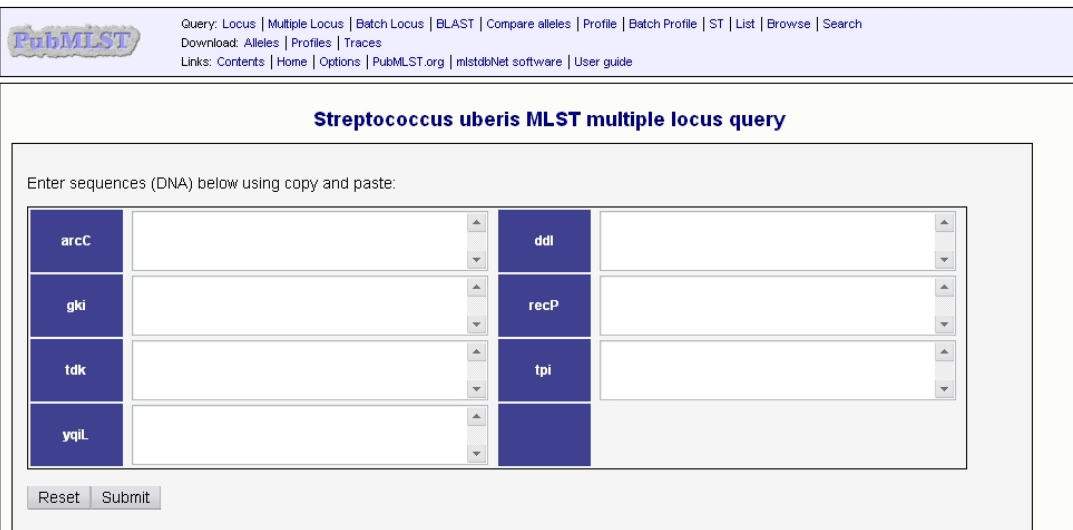
This part of a *MLST* analysis is about *DNA* isolation, purification and extraction of the desired gene fragments with *Polymerase chain reaction (PCR)*. Primers are available in most of the public databases (3.5). The preparation step supplies two sequences for each gene, the forward and the backward read.

3.7.2 Multilocus sequence typing

1. *Assembly*

The previous step produces two sequences (reads) for a gene with *PCR*. The

¹http://pubmlst.org/perl/mlstdbnet/mlstdbnet.pl?page=allseq&file=su_profiles.xml



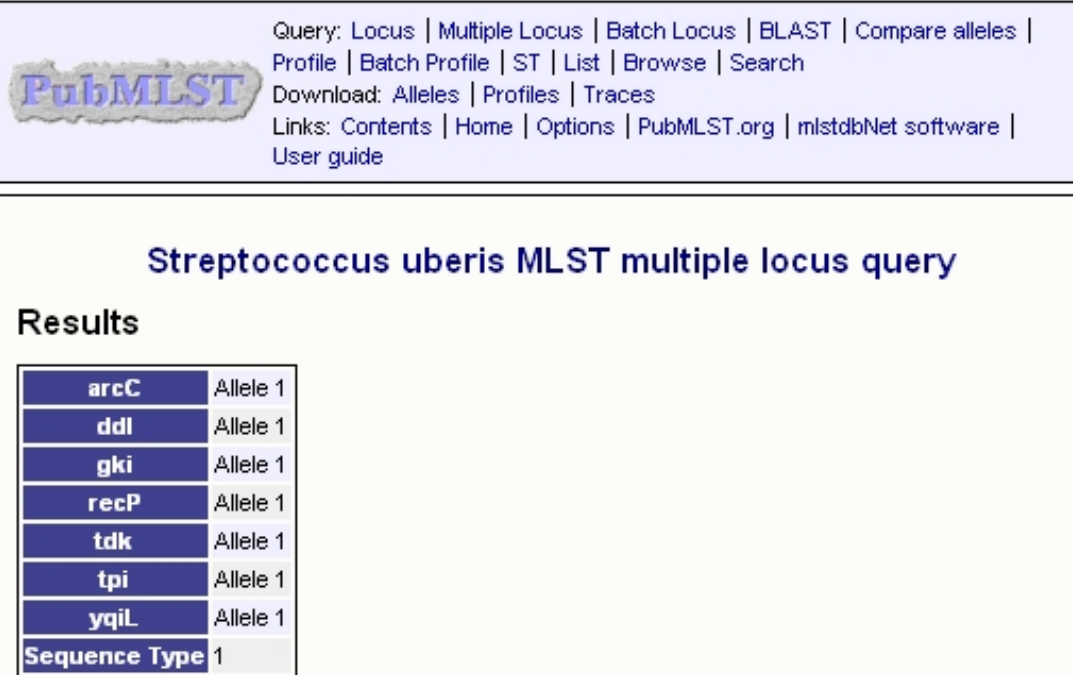
Query: [Locus](#) | [Multiple Locus](#) | [Batch Locus](#) | [BLAST](#) | [Compare alleles](#) | [Profile](#) | [Batch Profile](#) | [ST](#) | [List](#) | [Browse](#) | [Search](#)
Download: [Alleles](#) | [Profiles](#) | [Traces](#)
Links: [Contents](#) | [Home](#) | [Options](#) | [PubMLST.org](#) | [mlstdbNet software](#) | [User guide](#)

Streptococcus uberis MLST multiple locus query

Enter sequences (DNA) below using copy and paste:

arcC	<input type="text"/>	ddl	<input type="text"/>
gki	<input type="text"/>	recP	<input type="text"/>
tdk	<input type="text"/>	tpi	<input type="text"/>
yqiL	<input type="text"/>		

(a) Input form for *multilocus sequence typing* query of PubMlst [1].



Query: [Locus](#) | [Multiple Locus](#) | [Batch Locus](#) | [BLAST](#) | [Compare alleles](#) | [Profile](#) | [Batch Profile](#) | [ST](#) | [List](#) | [Browse](#) | [Search](#)
Download: [Alleles](#) | [Profiles](#) | [Traces](#)
Links: [Contents](#) | [Home](#) | [Options](#) | [PubMLST.org](#) | [mlstdbNet software](#) | [User guide](#)


Streptococcus uberis MLST multiple locus query

Results

arcC	Allele 1
ddl	Allele 1
gki	Allele 1
recP	Allele 1
tdk	Allele 1
tpi	Allele 1
yqiL	Allele 1
Sequence Type	1

(b) Result of a *multilocus sequence typing* query at PubMlst [1]. The sequences which were added in the input form are sequences of the *sequence type* 1.

Figure 5: Online available PubMlst's *sequence type* query for organism *Streptococcus uberis*: Figure 5a is showing the input form where you enter the sequences. Figure 5b is an example result. It is the query result if the input sequences are from a *sequence type* 1 isolate.



[Query: Locus](#) | [Multiple Locus](#) | [Batch Locus](#) | [BLAST](#) | [Compare alleles](#) | [Profile](#) | [Batch Profile](#) | [ST](#) | [List](#) | [Browse](#) | [Search](#)
[Download: Alleles](#) | [Profiles](#) | [Traces](#)
[Links: Contents](#) | [Home](#) | [Options](#) | [PubMLST.org](#) | [mlstdbNet software](#) | [User guide](#)

Streptococcus uberis MLST multiple locus query

Results

arcC	Allele 1
ddl	Allele 1
gki	Not found: nearest allele 1, 99.78% identity (View alignment) Differences: nt 169: A → T
recP	Allele 1
tdk	Allele 1
tpi	Allele 1
yqiL	Allele 1
Sequence Type	Incomplete data or no matches found

- (c) Result of a *multilocus sequence typing* query at PubMlst [1]. The sequences which were added in the input form are sequences of a supposable new *sequence type*. Gene “gki” was queried with a sequence which is not contained in the database and thus gets a new *allele type*. New *allele types* lead to supposable new *sequence types*.

Figure 5: (Continued) Online available PubMlst’s *sequence type* query for organism *Streptococcus uberis*: Figure 5c is an example result. It is an *isolate*, which did not match with a *sequence type* in the database and probably is a new *sequence type*.

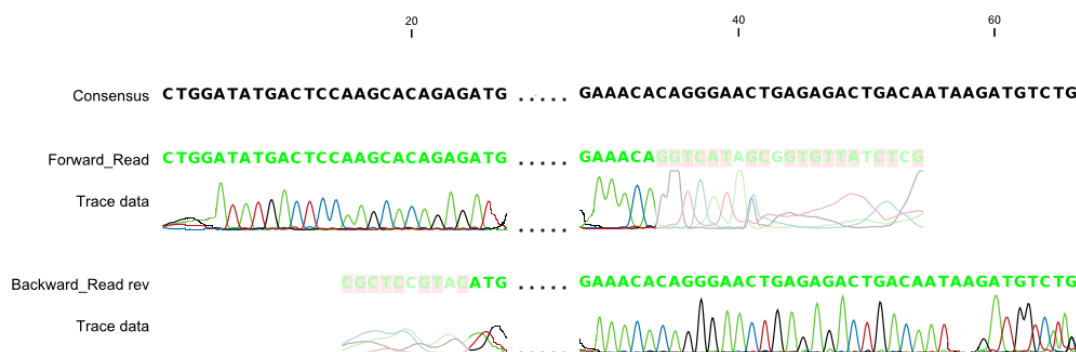


Figure 6: A contig: The *assembly* of two sequences (forward read and backward read) is shown. The result is the consensus sequence, which is used for further analyses.

forward and backward read of the *locus* is then assembled to get a consensus sequence for typing. Sometimes only one read, the forward or the backward read, could be extracted and is used for the *assembly*. The result of an *assembly* is the *contig*, which comprises the assembled sequences. Figure 6 shows an example of a *contig*.

2. Trim the consensus sequence
The resulting consensus sequence of the *assembly* has to be trimmed to the defined fragment length of the gene in order to type.
3. Query *allele type*
The trimmed consensus sequence is copied and pasted into an internet browser. For example the input form of the PubMlst databases [1] can be used to query the *allele type*, showed in Figure 5a.
4. Repeat for rest of genes
Step 1 to 3 have to be repeated for each gene. The result is a combination of numbers, the *allele types*.
5. Query *sequence type*
Use the discovered *allele types* and query the *sequence type* with for example the PubMlst databases [1].
6. Submission of new types
New *allele types* and *sequence types* should be submitted to a public database. Some databases are listed in Section 3.5.

3.7.3 Multilocus sequence analysis

The data collected and generated in the previous steps (Preparation, *Multilocus sequence typing*) are used to perform evolutionary studies. All analyses with this data is called *Multilocus sequence analysis*.

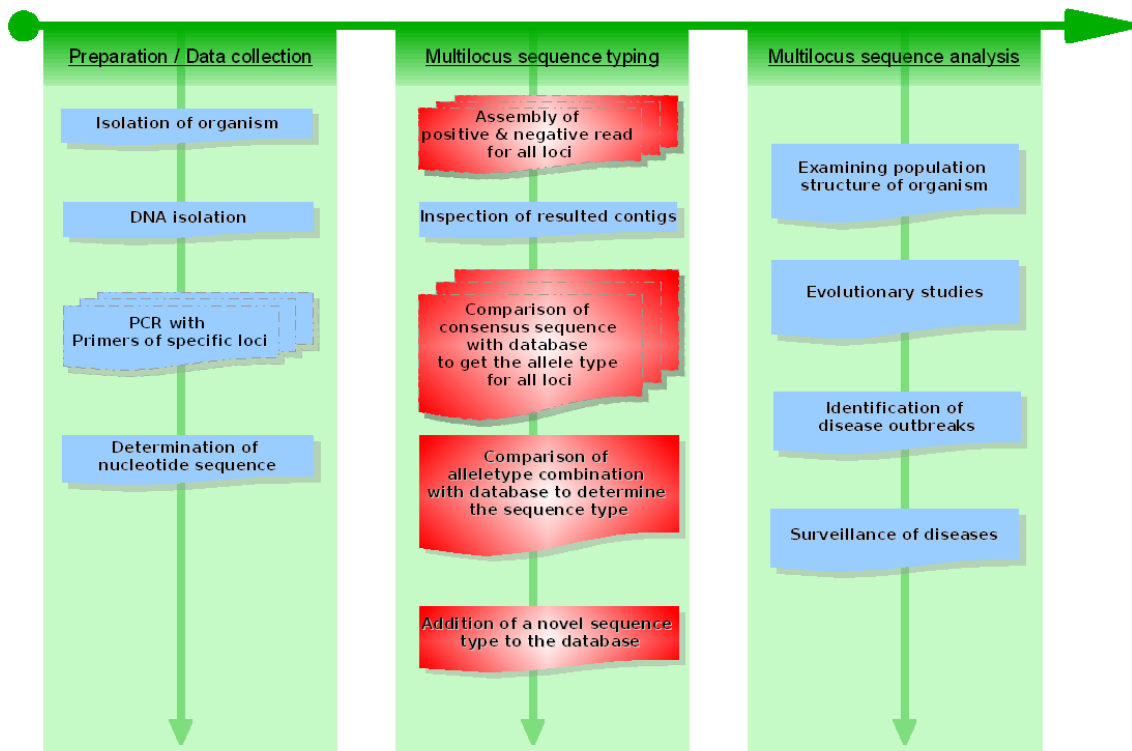


Figure 7: Overview of work regarding *MLST*. Contrary to Figure 2 the tasks are red which can easily be performed automatically.

4 Development

4.1 Motivation of the work

Multilocus sequence typing is a state-of-the-art technique in typing microbial organisms. Due to the fact that sequencing costs and time reduce severely, the *multilocus sequence typing* technique becomes more important and is adopted in several laboratories, because sequencing costs and time is the major constriction of this technique.

In Figure 2 the overall tasks in *MLST*, the preparation, *typing* and analyses afterwards are shown. This figure is also used to show the steps in *multilocus sequence typing*, which can easily be automated. As you see in Figure 7, the predominant part of the *MLST* box can be automated (red colour). Another aspect of the motivation of this work is that automation entails a reduction in time, which is a crucial factor in typing.

In Section 4.3 the few existing solutions are introduced. These solutions provide functionality for *multilocus sequence typing* but are not user-friendly and do not have a fast and easy workflow.

4.2 Aim of the work

The aim of this work is to implement a software module, which is user-friendly and intuitive in handling *MLST* data. A fast and automated workflow has to be designed for rapid *multilocus sequence typing* and the software shall be implemented in an extendable design. The following specifications are composed for the resulting software module:

- Advantages:
 - User-friendliness, intuitive user interface
 - Automated workflow
 - Persistence of data
 - Extendable software
 - *Cross-platform*
 - Support
- Disadvantages:
 - Expenses

The most disadvantages of the existing solutions (4.3) are planned to be avoided, however, expenses will be taken since this is a commercial product.

4.3 Existing solutions

This section examines the current available applications for the purpose of *multilocus sequence typing*. Disadvantages and advantages are elaborated.

4.3.1 Online form

The well-established databases are shown in Section 3.5. In order to type with those databases, it is possible to get information with several web-based forms, see the PubMlst databases [1, 21] at Page 11, as an example.

- Advantages
 - *Cross-platform*
 - No extra software installation
- Disadvantages:
 - Inexistent or badly arranged persistence of *MLST* data and results
 - Inexistent automation
 - Query by copy and paste of *MLST* data
 - Error-prone due to lots of user interactions (e.g. copy and paste)

4.3.2 Open source software / Freeware

Open source software and freeware stand out due to the fact that this software is free of charge. Unlike freeware, the code of an open source software is open to the public and thus it can be extended to specific requirements. Free software is for example the Staden Package [22, 23], START2 [24] and STARS [25].

- Advantages:
 - Free of charge
 - Open source software can be extended
 - Often persistence of data
- Disadvantages:
 - Mostly not *cross-platform*
 - Mostly not user-friendly designed, no intuitive user interface
 - Less or no support

4.3.3 Applied Biosystems

Applied Biosystems offers a workflow with using Applied Biosystems 3130 and 3730 Series Capillary Electrophoresis Systems and SeqScape[®] Software [26].

- Advantages:
 - Support
 - User-friendliness
 - Persistence of data
- Disadvantages:
 - Only available for platforms: Windows 2000, Windows XP
 - Expenses
 - Inflexible because it is not an extendable software
 - Manually query *sequence type* with an online form

4.4 End-user and product analysis

The software module will be addressed to users in laboratories who are dealing with molecular typing.

- Laboratories (Institutes, Reference laboratories)
- Hospitals
- Centers of surveillance

There are different cases where users have to make a decision about using software in order to type. The following cases demonstrate the main situations:

Case 1

Main focus of the research of an end-user is the *multilocus sequence analysis* of a specific organism. It is essential for the start of the research that the preparation (*DNA extraction*) and *multilocus sequence typing* has to be performed. Preparation and *multilocus sequence typing* requires time which can be reduced.

Case 2

Reference laboratories type more than hundred *isolates* per month. In order to keep track of the typed *isolates* there should be a software solution for enhancement. The *isolates* may be used in further analyses.

Case 3

Public available databases don't offer a *MLST scheme* of the organism, desired for *typing*.

Case 4

Public available databases contain the desired *MLST scheme* of the organism, but the end-user doesn't want to publish the results (upload to the database). Nevertheless the user expects immediate and updated results.

Case 5

The end-user wants to edit a *MLST scheme* in order to fit his/her needs.

All these cases show that there is a need for a flexible software solution.

4.5 Software development process

The procedure of the software development process is defined in this section. Used techniques and new approaches are introduced.

4.5.1 Feature Driven Development (FDD)

The *Feature Driven Development (FDD)* is a iterative software development process and therefore an agile method. First process of a *FDD* approach is "Develop an overall model". This process produces an overall object model which represents more shape than definitive content. The next step is "Build features list". In this step, a list is generated of all features, which should be implemented. The third and last process in the startup phase is the "Planning" process. This process supplies a plan for the upcoming development. The startup phase follows the construction phase. This phase consists of iterations for every feature in the feature list, processes

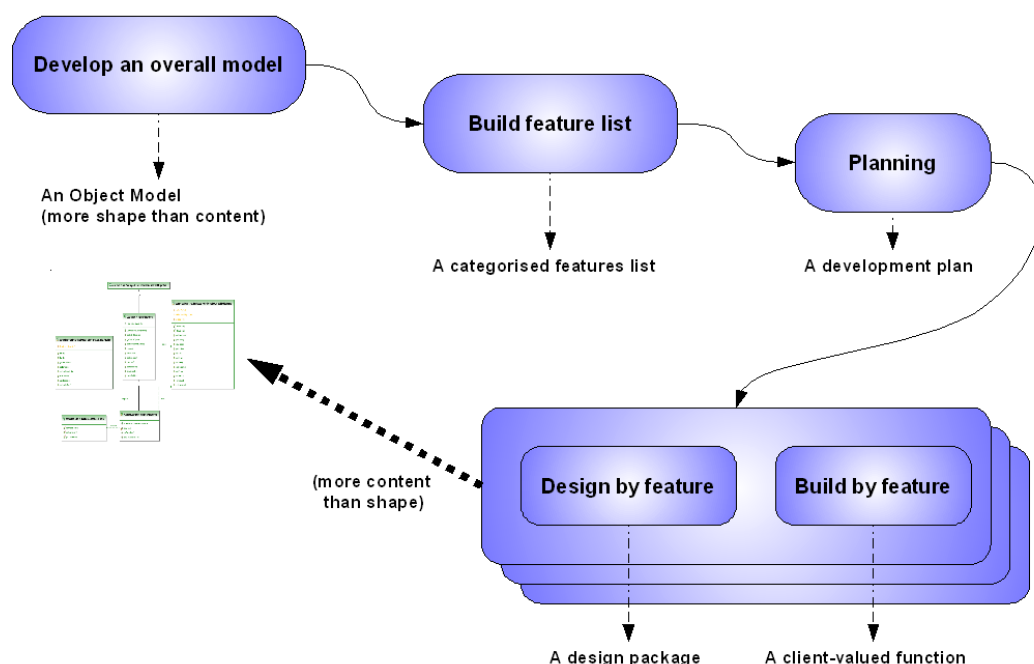


Figure 8: The Feature Driven Development software process. Source: Nebulon Pty. Ltd.

“Design by feature” and “Build by feature”. The processes are shown in Figure 8.

In Section 5 (Requirements specification), the overall model and features list of this system are explained. The prioritizing of this feature list, described in Section 5.6.1, arise from the “Planning” process. All feature’s design and development is summarized in the Section 6.

4.5.2 Test Driven Development (TDD)

A *Test Driven Development* is a technique in software development. A cycle of implementing a new feature or new code begins with implementing tests, which fail, because the code is not yet implemented. After finishing the tests, the implementation of the feature starts. The feature can only be accepted, when the tests succeed and the feature is then properly implemented. At the end of the cycle the code is refactored, but the tests still have to succeed. When a new cycle was implemented, all tests have to pass!

4.5.3 FDD-TDD approach

The implementation of the module is a *FDD* with *Test Driven Development (TDD)*. In Figure 9, you can see a part of the *FDD* approach, the “Build by feature” part. This part is extended with *TDD* in this project.

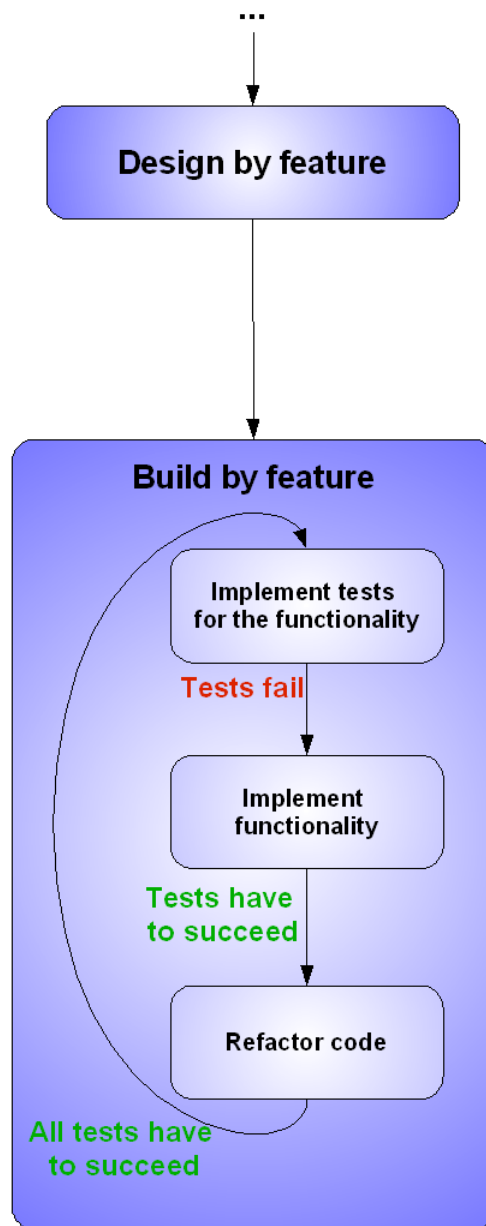


Figure 9: *FDD-TDD* approach of this project. The “Build by feature” process of *Feature Driven Development* is extended with *Test Driven Development*. Every feature implementation starts with implementing tests. After finishing the tests, the functionality can be implemented. The tests have to succeed, before passing over to refactoring. The “Build by feature” step is iterative, that means the feature can be partitioned before the implementation starts.

4.6 Java

The programming language *Java* [27] was chosen in order to develop with the CLC Developer Kit (Section 4.8), which is also implemented with *Java*. This programming language is object-oriented, that means programming deals with creating objects, manipulating objects and make objects work together. *Java* is an easy to use programming language, since *Java* has an automatic memory allocation and garbage collection. Advantages of *Java* are the *cross-platform* feature (a cross-platform program can be used on different platforms, like Windows or Linux), as well as the characteristic of multithreaded implementations. All the elaborated abilities of *Java* turn it to a perfect programming language for this project.

The *Java* Standard Edition (Java SE) version 1.4 is used to implement the module. This version is used in order to keep the *cross-platform* feature since there are compatibility problems with Mac OS X versions and newer *Java* versions.

4.7 JUnit

As explained in Section 4.5.3 the software development process includes a *Test Driven Development* approach. In order to start testing the planned features written in *Java*, the *JUnit* framework is used.

JUnit is a *Java*-based framework for writing and automatically executing unit tests. Unit testing means, testing parts of the program. The size of the unit, that is intended to be tested, varies from methods to classes to components.

4.8 CLC bio's Software Developer Kit

The Software Developer Kit of CLC bio is written in *Java* and provides easy to use frameworks for establishing algorithms and user interfaces. It is used to implement plug-ins for the existing software, the CLC Workbenches see Figure 1.

This project is the first module, that is implemented with the CLC Developer Kit.

5 Requirements specification

5.1 Purpose

This specification describes the requirements for the plug-in *CLC MLST Module*. This plug-in is intended to work in combination with the CLC Workbenches. The *CLC MLST Module* will offer a fast workflow for typing microorganisms with *Multilocus sequence typing*. Furthermore, the *CLC MLST Module* will provide an easy way of creating, editing and managing *MLST* related data.

5.2 Scope

In this section the scope of the software is described. The features within the scope are developed, whereas features out of the scope are not.

In scope

- Typing workflow
- Manipulating *MLST scheme* data
- Manipulating *Isolate* data
- Downloading *MLST schemes*

Out of scope

- Functionality for *multilocus sequence analysis*

5.3 Product perspective

The project is a new plug-in in the product family of upcoming molecular diagnostics software for CLC bio [6]. This plug-in will only be used in combination with one of the existing software solutions of CLC bio, the CLC Workbenches (Figure 1).

5.4 Product features

This section represents the overall structure of product features which will be more elaborated in Section 5.6. Figure 10 shows a data flow diagram of the general features.

MLST data will be data regarding *isolates* and *MLST schemes*. Therefore the “MLST data” term represents both, *isolate* data and *MLST scheme* data.

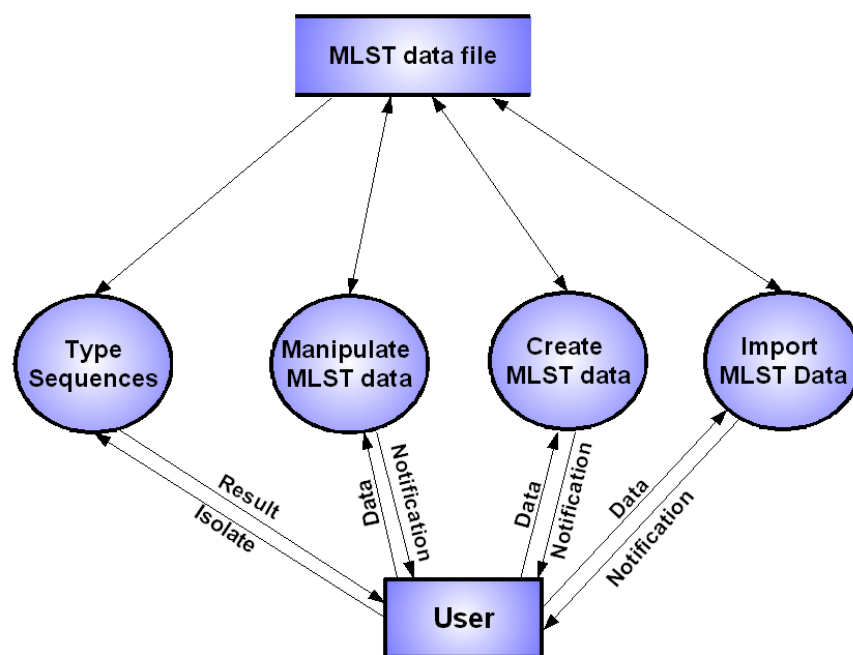


Figure 10: A data flow diagram, representing the overall features, which will be integrated in the *CLC MLST Module*. Arrows represent data flow, circles imply features. The “User” is an interface and the “MLST data file” the data storage.

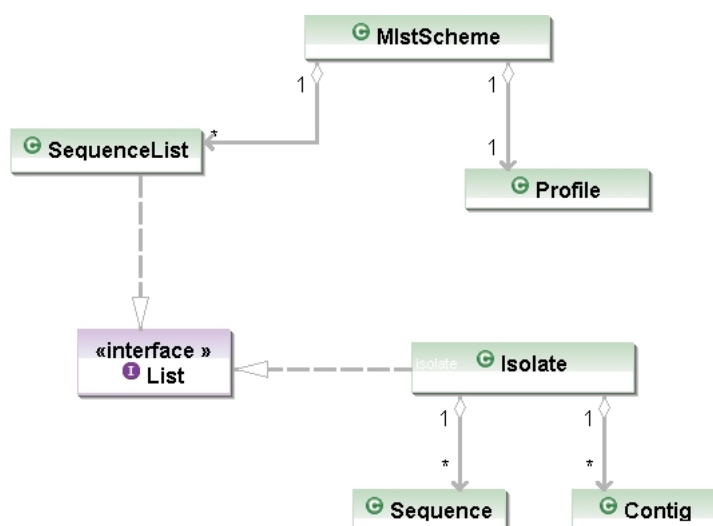


Figure 11: An overall class diagram of the most important classes in the *CLC MLST Module*

5.5 User classes and characteristics

The real world is interpreted into an object-oriented model here. The following classes don't represent software classes, they represent a concept.

5.5.1 Overview

The Figure 11 shows the overall class concept for the *CLC MLST Module*. These classes are a view of the objects, handled with *MLST*, and do not provide *Java* classes.

5.5.2 Detailed model

This section describes the model introduced in Section 5.5.1 in detail. The responsibilities and attributes are stated.

5.5.2.1 MLST scheme

The responsibility of this class is the maintenance of the *allele* sequences and the *profiles*. It is in charge of additions of no redundant entries, since redundancy is forbidden in *MLST schemes*. Attributes are lists of sequences for every gene, *profiles*, comprising all entries for the *sequence types*, and a version number.

5.5.2.2 SequenceList

This list takes care of sequences. Attributes are the sequences.

5.5.2.3 Profile

The “*Profile*” class is responsible for the addition and deletion of new number combinations in *profiles*. It is again prohibited to have redundant lines in *profiles* of an *MLST scheme*.

The “*Profile*” class represents the number combination for every *sequence type*. An example of a *profile* is shown in Figure 3b. In addition to the number combinations of the *sequence types*, the “*Profile*” class also contains the header (the name of the genes and any additional information).

5.5.2.4 Isolate

The “*Isolate*” class takes care of the sequences which will be typed. Those sequences can be assembled into a *contig* or the sequence itself will be taken for typing. So, the “*Isolate*” object can be composed of *contigs* or sequences depending on the number of genes. The “*Isolate*” class is handling the *typing*, that means that it knows the “*MLST scheme*” object, to type against. The “*Isolate*” class has to have the same number and names of genes as its referenced “*MLST scheme*”, otherwise it would be not clear how to type. The “*Isolate*” can be in a status, where not all genes have a sequence or *contig* assigned.

Attributes are the sequences or the *contigs* for every gene and a reference to a “*MLST scheme*”. The “*Isolate*” class contains the result of the *typing*, the numbers of the eligible *sequence types*. Additional it contains a report which collects all the results concerning the *typing*.

5.5.2.5 Contig

The “*Contig*” class comprises the sequences, which are assembled together. This class also encloses a consensus sequence.

Attributes are the sequences, the location of trimming and the resulting consensus sequence.

5.5.2.6 Sequence

The “*Sequence*” class represents a sequence. Attributes are the sequence and annotations.

5.6 System features

This section illustrates the system features of the product. A feature will be described and prioritized, the stimulus or response sequence is elaborated (written and use case diagram) and the functional requirements, the features of the software,

are stated.

5.6.1 Features list

A feature set will be divided into subfeatures, the functional requirements. The following list identifies feature sets of the system and states them in prioritized order. After this section the individual feature sets are presented and described in detail.

1. Type *isolates* with *MLST scheme*
2. Create *MLST scheme*
3. Download *MLST scheme*
4. Merge *MLST schemes*
5. Extend *MLST scheme* with sequences or *isolates*
6. Extend *isolate* with sequence or *contig*
7. Submit *isolate* to a public database

5.6.2 Typing workflow

5.6.2.1 Description and Priority

This feature enables the typing of sequences with *multilocus sequence typing*. The user has sequence files for typing.

This feature is the main feature of the product and therefore of highest priority.

5.6.2.2 Stimulus/Response Sequences

1. The user imports sequence files into a CLC Workbench
2. The user executes an action for *typing* with *MLST*
3. The system brings up a window
4. The user selects the sequences for *typing*
5. The user verifies the selection
6. Parameters for the *assembly* and the *MLST scheme* have to be set by the user and verified
7. The system offers an automatic assignment of the selected sequences to the genes of the selected *MLST scheme*
8. The user verifies or changes the automatic assignment
9. The user finishes the window

10. The system signals the *typing* result to the user
11. The system creates an object (*Isolate*) containing the sequences assigned to the genes and the typing result.

The use case diagram in Figure 12 demonstrates the use case “*Typing workflow*”.

5.6.2.3 Functional Requirements

Requirement 1 The system shall offer an import functionality for sequences

Requirement 2 The system shall have a “Typing” action

Requirement 3 The system has to have an automatic assignment option

Requirement 4 The system offers a form for user inputs

Requirement 5 The system shall be able to type sequences against a *MLST scheme*

Requirement 6 The system shall create an object containing the sequences and the typing results

Requirement 7 If the user enters invalid input the next step cannot be entered or the action cannot be finished

5.6.3 Create MLST scheme

5.6.3.1 Description and Priority

It is absolutely essential for typing that an *MLST scheme* is used. *MLST schemes* have become standard and can be exchanged with different laboratories. It is not possible to create new *MLST schemes* with public databases and therefore this is a high priority feature.

5.6.3.2 Stimulus/Response Sequences

1. The user executes an action associated with the feature “Create MLST scheme”
2. The system opens a dialog
3. The user enters gene names which shall be contained in the *MLST scheme*
4. The user verifies the input
5. If the user has *allele* sequences to include into the *MLST scheme*, the sequences can be added.

The use case diagram, Figure 13, demonstrates the use case “Create *MLST scheme*”.

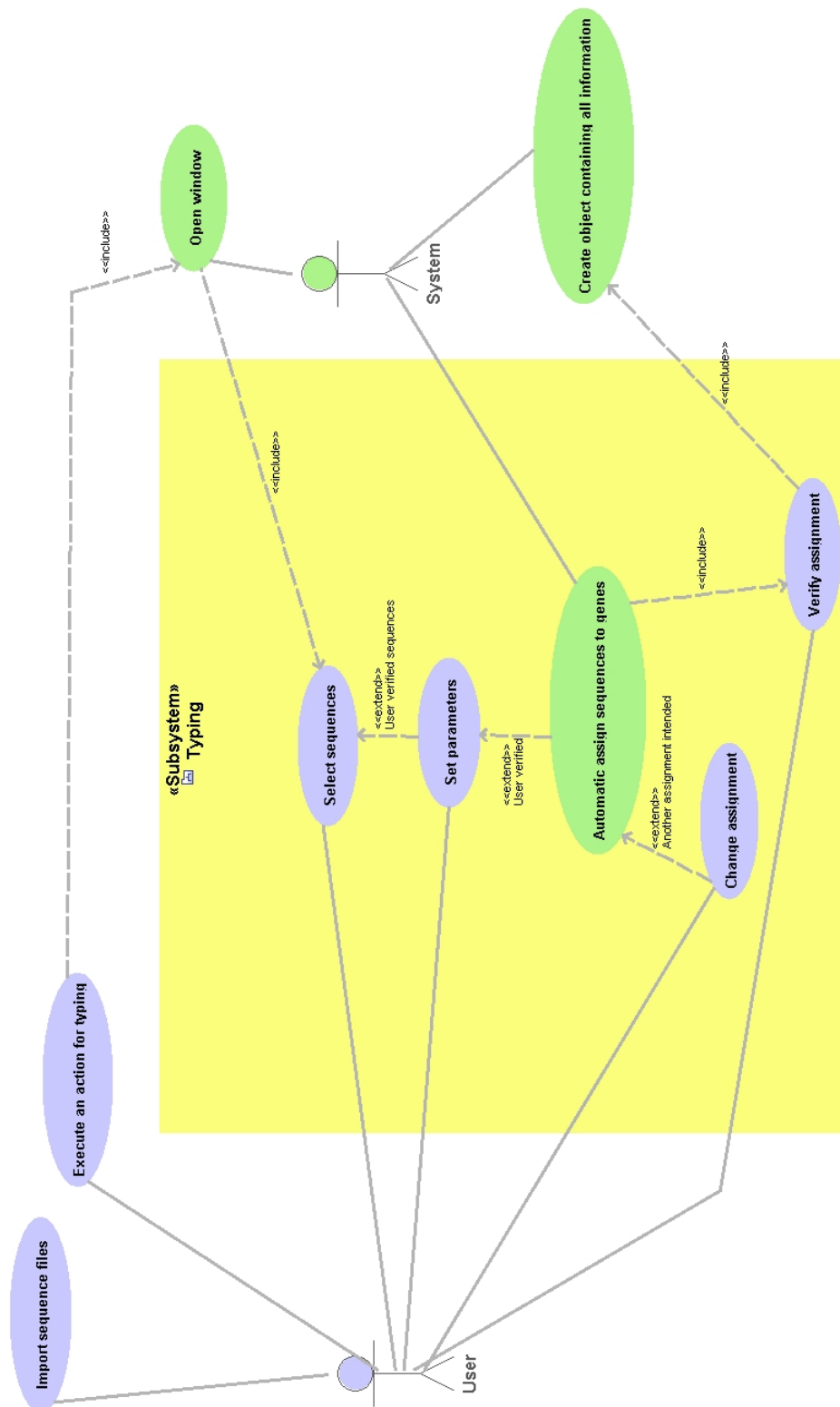


Figure 12: Use case of system feature “Typing workflow”. Main actors are the user and the system. The subsystem ‘Typing’ represents the main functionality for *typing*.

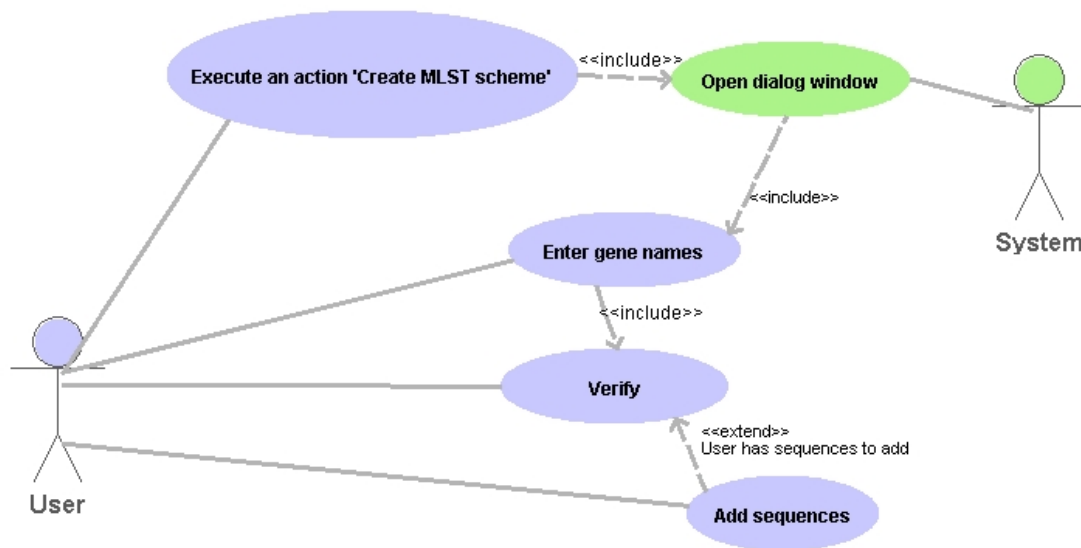
Create MLST scheme

Figure 13: Use case of system feature “Create *MLST scheme*”.

5.6.3.3 Functional Requirements

Requirement 1 The system shall have an action for creating new *MLST schemes*

Requirement 2 The system shall offer a form for user inputs, the gene names

Requirement 3 The system shall offer a form for adding sequences

Requirement 4 If the user enters invalid input the next step cannot be entered or the action cannot be finished

5.6.4 Download MLST scheme**5.6.4.1 Description and Priority**

The user shall have the option to download an existing *MLST scheme* to work with, means to type, and to extend and change it. Public databases contain several *MLST schemes*, see in Section 3.5. This feature is of high priority.

5.6.4.2 Stimulus/Response Sequences

1. The user executes an action for downloading *MLST schemes*
2. The system opens a dialog window
3. The system offers organisms which are available for the download
4. The user selects the desired organisms
5. The system downloads and saves the selected *MLST schemes*

The use case diagram, Figure 14, demonstrates the use case “Download *MLST scheme*”.

Download *MLST scheme(s)*

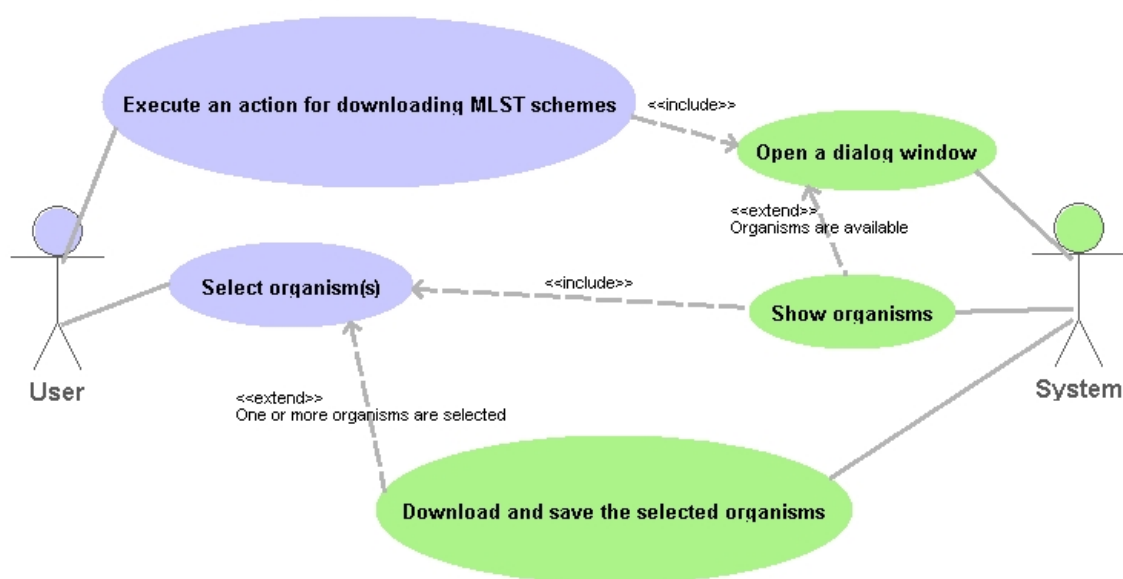


Figure 14: Use case of system feature “Download *MLST scheme*”.

5.6.4.3 Functional Requirements

Requirement 1 The system shall have an action concerning download of *MLST schemes*

Requirement 2 The system downloads the names of the available organisms

Requirement 3 The system offers a dialog to show the organisms and where the user is able to set parameters (where to save)

Requirement 4 The system shall download *MLST schemes*

Requirement 5 The system shall save *MLST schemes*

5.6.5 Merge *MLST schemes*

5.6.5.1 Description and Priority

The merge of *MLST schemes* is required for summarizing different *MLST schemes*. If there are two or more *MLST schemes*, it is then possible to create a common *MLST scheme*, comprising all information. This merged *MLST scheme* is then used for typing. For instance, it is necessary to merge *MLST schemes* for using an updated version. In this situation an updated version is a version of an *MLST scheme*, that was downloaded from a public database and extended locally. If there

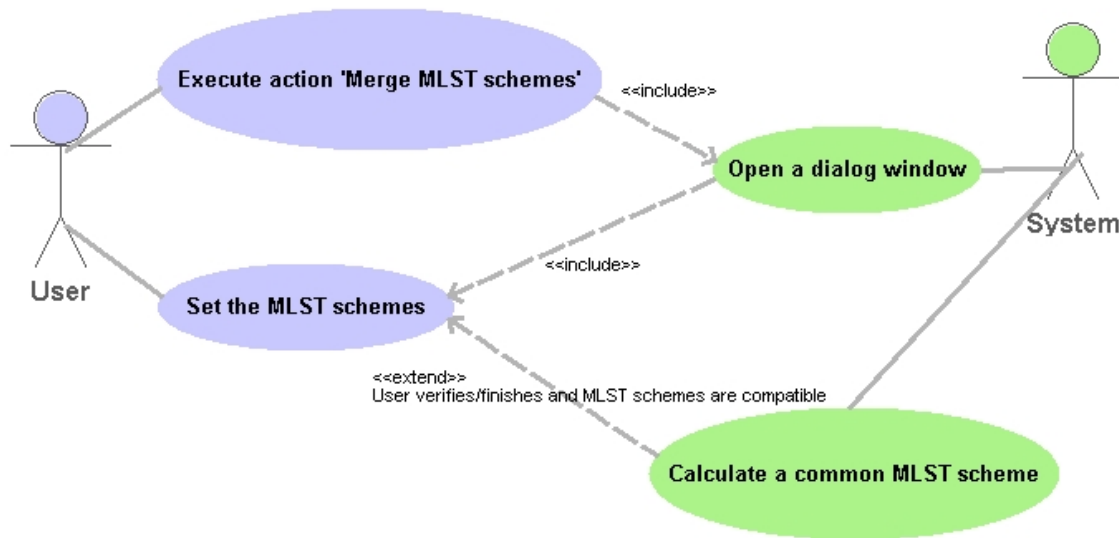
Merge MLST schemes

Figure 15: Use case of system feature “Merge *MLST schemes*”.

is a new version of the downloaded *MLST scheme* available, the user wants to merge local changes with changes of the public database.

This feature is of high priority.

5.6.5.2 Stimulus/Response Sequences

1. Execute an action associated with merging *MLST schemes*
2. The system opens a dialog window
3. Set the *MLST schemes* to merge
4. Verify the selection and finish the action
5. The system calculates a common *MLST scheme*
6. If the selected *MLST schemes* are not compatible, that means they belong to different organisms or have different genes, the calculation will not be executed.

The following use case diagram, Figure 15, demonstrates the use case “Merge *MLST schemes*”.

5.6.5.3 Functional Requirements

Requirement 1 The system shall have an action for merging *MLST schemes*

Requirement 2 The system shall have a dialog where the user can set the *MLST*

schemes and other parameters

Requirement 3 The system shall have an algorithm merging two or more *MLST schemes*

Requirement 4 The system shall identify not compatible *MLST schemes*

5.6.6 Extend MLST scheme

5.6.6.1 Description and Priority

An *MLST scheme* contains entries for *sequence types*. When an *isolate* is typed, it occurs that a *sequence type* is assigned or that the *isolate* is a new type. In the case of a new type, the user wants to include the new type into the *MLST scheme*. The feature of extending *MLST schemes* handles the addition of new *sequence types*, in order to have an updated version of a *MLST scheme*. This system feature has a high priority.

5.6.6.2 Stimulus/Response Sequences

The user has sequences, which represent a new *sequence type* after he/she typed the sequences against an *MLST scheme*.

1. The user executes an action which enables adding of new *sequence types* to a *MLST scheme*
2. The system opens a dialog window
3. The user selects the *isolates*
4. The user verifies the selection
5. The system adds the *isolates* which are new types to the *MLST scheme*
6. The system updates the *isolates* with the new version of the *MLST scheme*

The diagram in Figure 16 demonstrates the use case “Extend *MLST scheme*”.

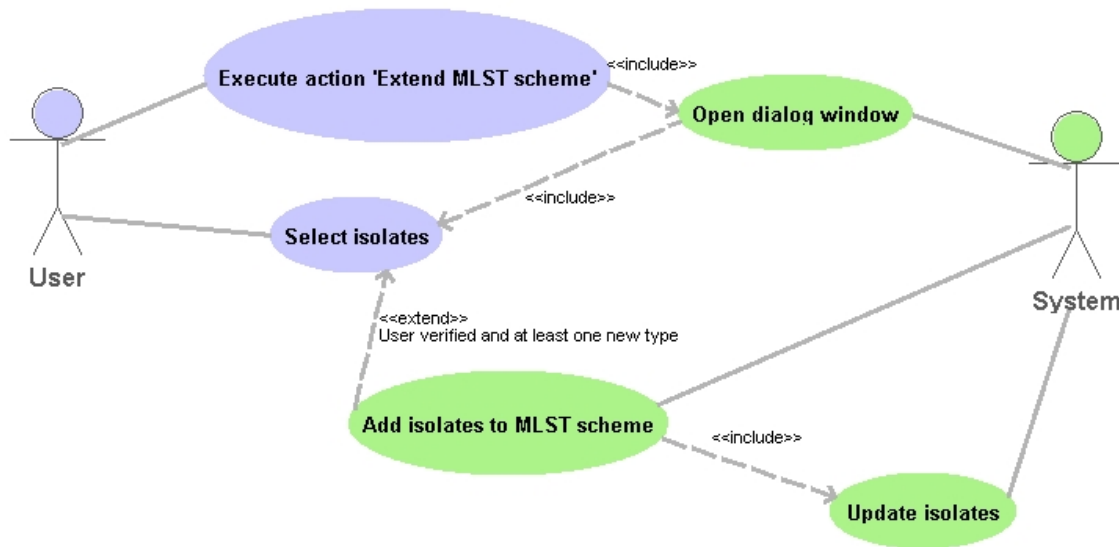
5.6.6.3 Functional Requirements

Requirement 1 The system shall have an action associated with extending *MLST schemes*

Requirement 2 The system shall have a dialog window, where the user can select the isolates and set parameters

Requirement 3 The system shall have an algorithm, which adds *isolates* as *sequence types* to an *MLST scheme*

Requirement 4 The system shall distinguish between new and defined *sequence types* as *isolates*

Extend MLST scheme**Figure 16:** Use case of system feature “Extend MLST scheme”

Requirement 5 The system shall have an update functionality, which updates an *isolate* with respect to its associated *MLST scheme*. Update an *isolate* means to type the *isolate*.

5.6.7 Extend Isolate**5.6.7.1 Description and Priority**

The *Isolate* class will have the responsibility of managing the sequences and the typing result. The *Isolate* class shall be able to contain different forms of sequences (sequences itself or *contigs*). The number of sequences or *contigs* in the *Isolate* depends on the number of genes of the associated *MLST scheme*. The user is able to create an *isolate* with the feature “*Typing workflow*”, however, the *isolate* doesn’t have to be fully assigned. That means, that the *isolate* has not recognized or not assigned *allele types* causing a higher number of possible *sequence types*. After the creation of an *isolate* the user is able to delete already assigned sequences or *contigs* and extend the *isolate* with other data.

For instance a laboratory types 100 *isolates* and starts with preparing the data collection of the first gene of all 100 *isolates*. The laboratory is then already able to create the *Isolates* objects in the software only with the first sequenced gene of each *isolate*.

This feature is of high priority.

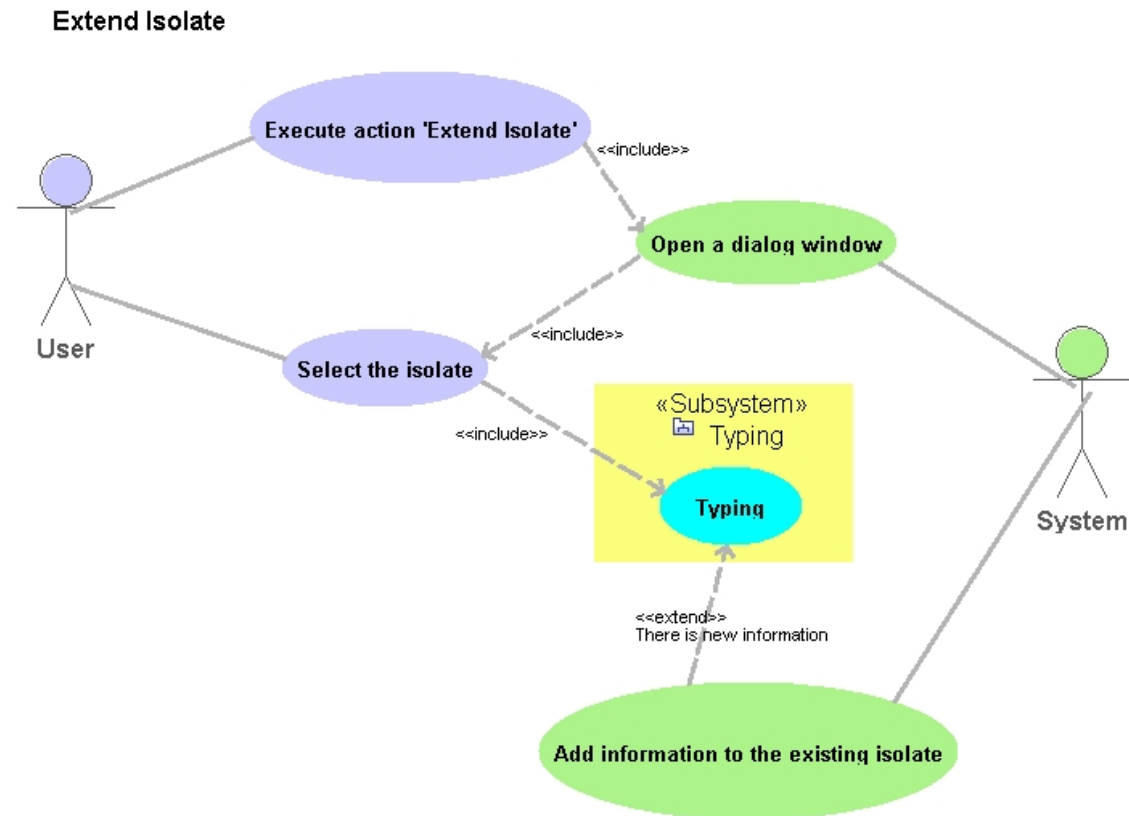


Figure 17: Use case of system feature “Extend Isolate”. The subsystem “*Typing*” of the system feature “*Typing workflow*” in Section 5.6.2, Page 29, is reused.

5.6.7.2 Stimulus/Response Sequences

1. The user executes an action associated with the extension of an *isolate*
2. The system opens a dialog window
3. The user has to select an *isolate*
4. The system performs the “*Typing*” feature, see in Section 5.6.2. The subsystem “*Typing*” in the Figure 12, Page 31, is executed.
5. The system adds the information to the existing *isolate*

The use case diagram in Figure 17 demonstrates the use case “Extend *Isolate*”.

5.6.7.3 Functional Requirements

Requirement 1 The system shall have an action associated with the extension of *isolates*

Requirement 2 The system offers an input form for parameters and a dialog window

Requirement 3 The system has an algorithm for typing

Requirement 4 The system shall add information to existing *Isolate* objects

5.6.8 Submission of data

5.6.8.1 Description and Priority

The profit of public databases is the aggregation of data around the world. In order to keep those databases up-to-date, it is beneficial that a lot of users submit their data. In order to automate the submission, the system feature “Submission of data” is introduced and prioritized with a medium level.

5.6.8.2 Stimulus/Response Sequences

1. The user selects the data which shall be submitted
2. The user executes an action associated with the submission of data
3. The system opens a dialog window
4. The user enters additional information (for example name, institute, source of *isolate*, ...)
5. The system submits the data with additional information to a public database

The following use case diagram, Figure 18, demonstrates the use case “Submission of data”.

5.6.8.3 Functional Requirements

Requirement 1 The system shall have an action associated with the submission of data

Requirement 2 The system shall offer a dialog window for user input

Requirement 3 The system shall have an algorithm for the submission of data

5.7 Non-functional requirements

In this section the non-functional requirements are elaborated. Non-functional requirements are quality attributes of the software rather than specific behaviors which is elaborated in Section 5.6.

5.7.1 Performance requirements

The performance for the system feature “Typing workflow” will be under 1 second for an *MLST scheme* described in Section 3.4. All other system features will vary in the performance, however, the user shall be notified immediately with a progress

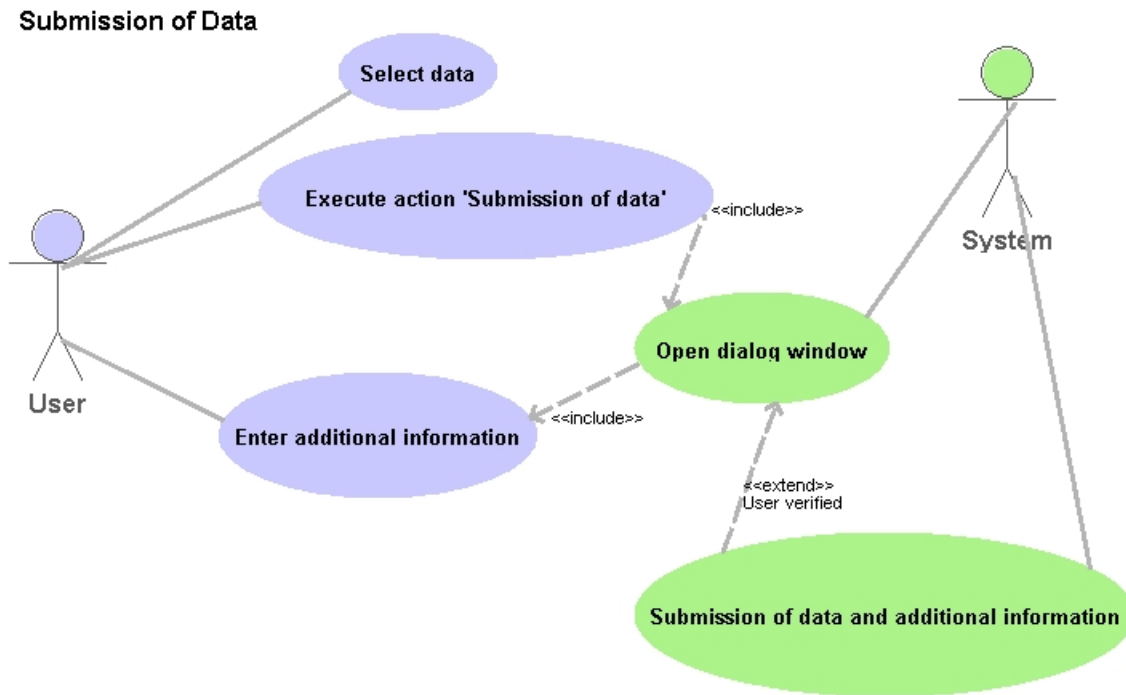


Figure 18: Use case of system feature "Submission of data".

bar.

5.7.2 Software Quality Attributes

The software shall be self-explained in order to ease the use and this should result in high usability. The user expects a software *typing* correct, so reliability in *typing* is also of high importance for the consumer acceptance. The automatic *typing* workflow shall be as reliable as possible. The software shall also be able to be easily integrated into the existing workflow of users.

From the view of the developer the maintainability is important, since the software will be maintained and extended by other developers.

6 Realization/Implementation

The realization of the specification is described in detail in this section. For some features there are utilities of the company CLC bio [6]. These utilities are mentioned in order to differentiate the work of this thesis and the existing software.

6.1 Persistence

The fundamental requirement for *MLST* is the persistence of the data acquired during typing. For this purpose, the concept of the classes in Section 5.5 is translated

into a *Java* concept, demonstrated in Figure 19.

The persistence is handled with serialization of *Java* objects. Serialization is the process of saving an object to a storage medium. The framework for saving the objects has already been implemented, therefore the objects of the *CLC MLST Module* only had to implement the existing interfaces. A database approach was not introduced with the *CLC MLST Module* project, since the database approach was implemented by other developers at the same time.

6.2 System features

6.2.1 Typing workflow

The workflow regarding *Multilocus sequence typing* starts with the import of sequences of an organism. Those sequences are the basis of the typing. The functionality of this import is available with the existing software.

The assembly functionality is also available with the existing software and is therefore reused. The “Assembly to reference” algorithm of the CLC bio software is used, in order to assign automatic and to calculate the *contigs*. This algorithm tries to align the input sequences to a reference sequence and the result is a *contig*. The consensus sequence of this *contig* is used to type against an *MLST scheme*.

Next, the functional requirements of Section 5.6.2 are described with the realized implementation.

Requirement 1 THE SYSTEM SHALL OFFER AN IMPORT FUNCTIONALITY FOR SEQUENCES

This requirement was already implemented with the existing software

Requirement 2 THE SYSTEM SHALL HAVE A “TYPING” ACTION

The action is called “Assembly and create *Isolate*”, since the sequences are assembled and the result is an *Isolate* object.

Requirement 3 THE SYSTEM HAS TO HAVE AN AUTOMATIC ASSIGNMENT OPTION

In Figure 23c (Page 46), the dialog shows the automatic assignment for gene “arcC”. The left list of sequences couldn’t be aligned to the reference sequence (first *allele* sequence of the gene “arcC” in the associated *MLST scheme*), and therefore the sequences in the left list were not assigned. Only the sequence “arcC_1_F” and “arcC_1_R” were able to be assigned. In the next steps of the dialog the other genes are handled. The activity diagram in Figure 20 demonstrates the automatic assignment. An activity diagram shows the step-by-step operations of a system.

Requirement 4 THE SYSTEM OFFERS A FORM FOR USER INPUTS

The input step is shown in Figure 23b (Page 45). The *MLST scheme*, alignment options and trimming options can be set. Furthermore, information is shown that the user can finish now (only if the parameters are valid, e.g. the

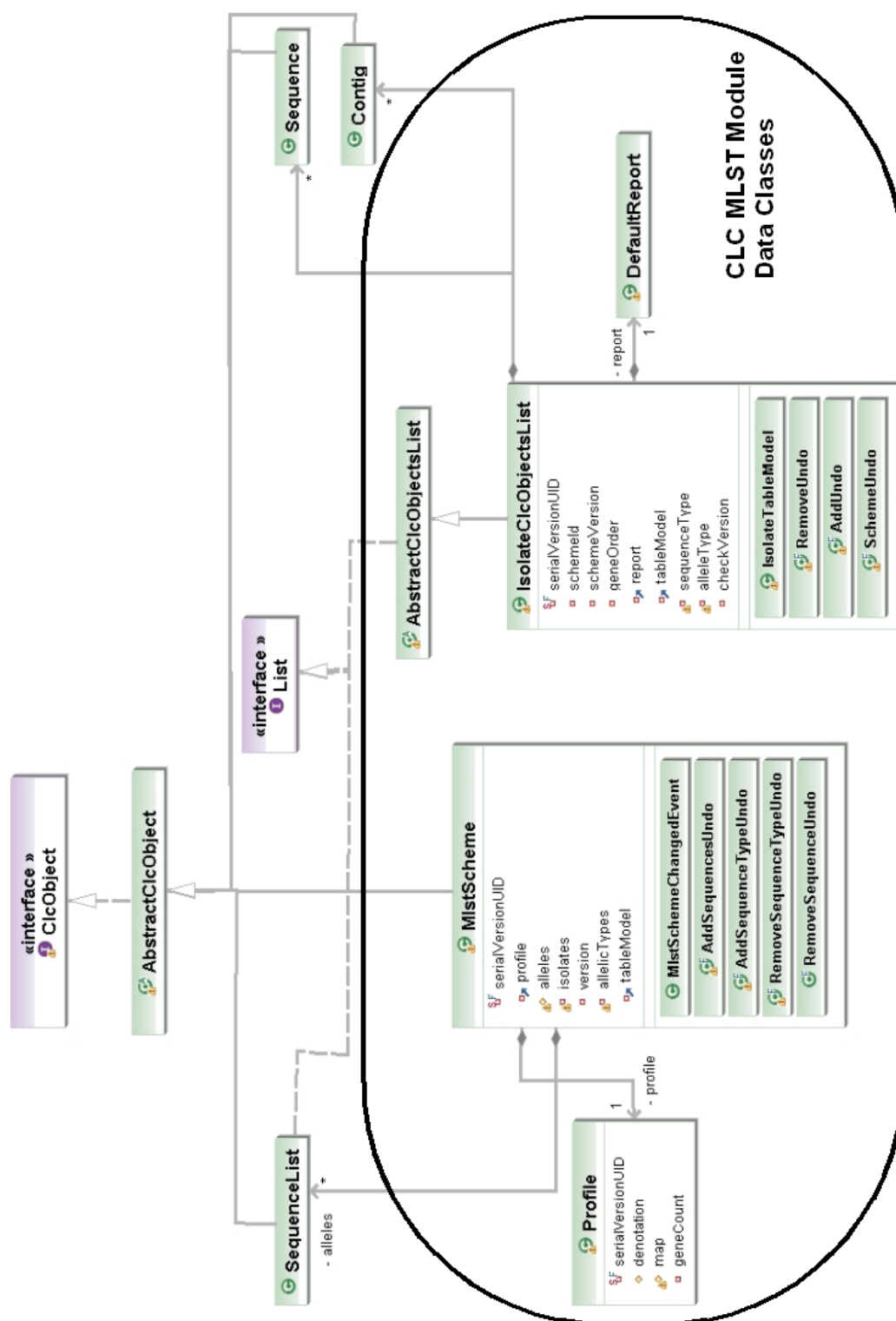


Figure 19: Data classes introduced with the MLST module. Some classes like the *Contig* and the *SequenceList* have already been introduced with the CLC bio software and are reused for the MLST module project.

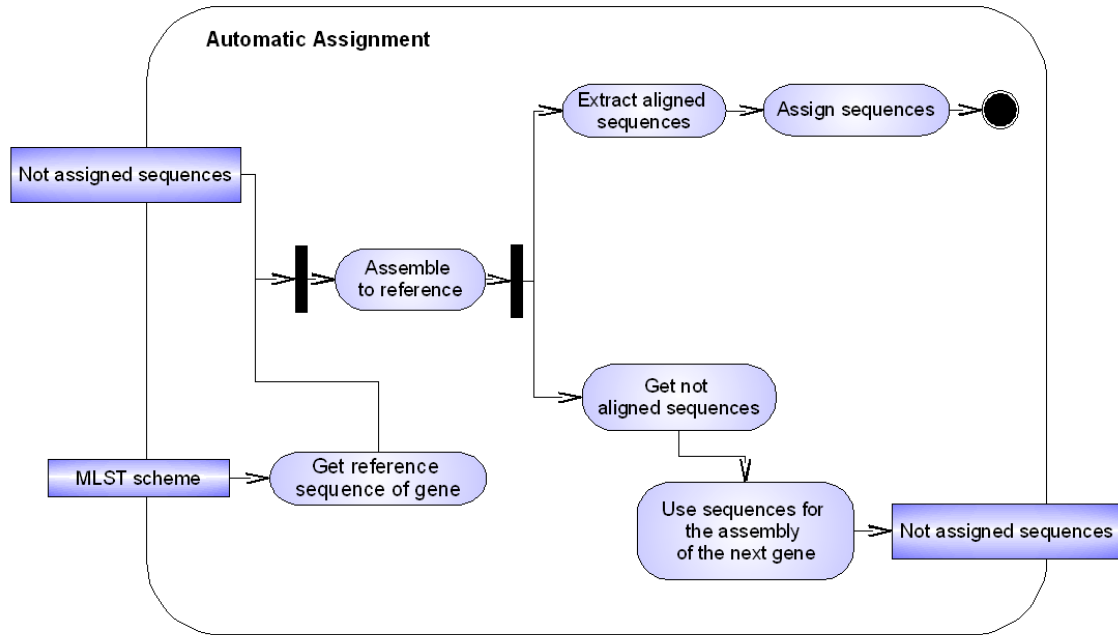


Figure 20: Activity diagram of the automatic assignment

MLST scheme is set) and the automatic assignment will assign the sequences to the genes.

Requirement 5 THE SYSTEM SHALL BE ABLE TO TYPE SEQUENCES AGAINST AN *MLST scheme*

The algorithm for typing sequences against an *MLST scheme* was implemented. The activity diagrams in Figures 21, 22 visualise the *typing* algorithm.

Requirement 6 THE SYSTEM SHALL CREATE AN OBJECT CONTAINING THE SEQUENCES AND THE TYPING RESULTS

The system creates the *Isolate* object, filled with the result information. In Figure 23e (Page 47), the *Isolate* object is shown with its editor, described in Section 6.4 (Page 55).

Requirement 7 IF THE USER ENTERS INVALID INPUT THE NEXT STEP CANNOT BE ENTERED OR THE ACTION CANNOT BE FINISHED

This is realized with the dialog windows of the action “Assembly and Create *Isolate*”. The “Next” and “Finish” button (e.g. in Figure 23) is enabled or disabled according to the entered parameters.

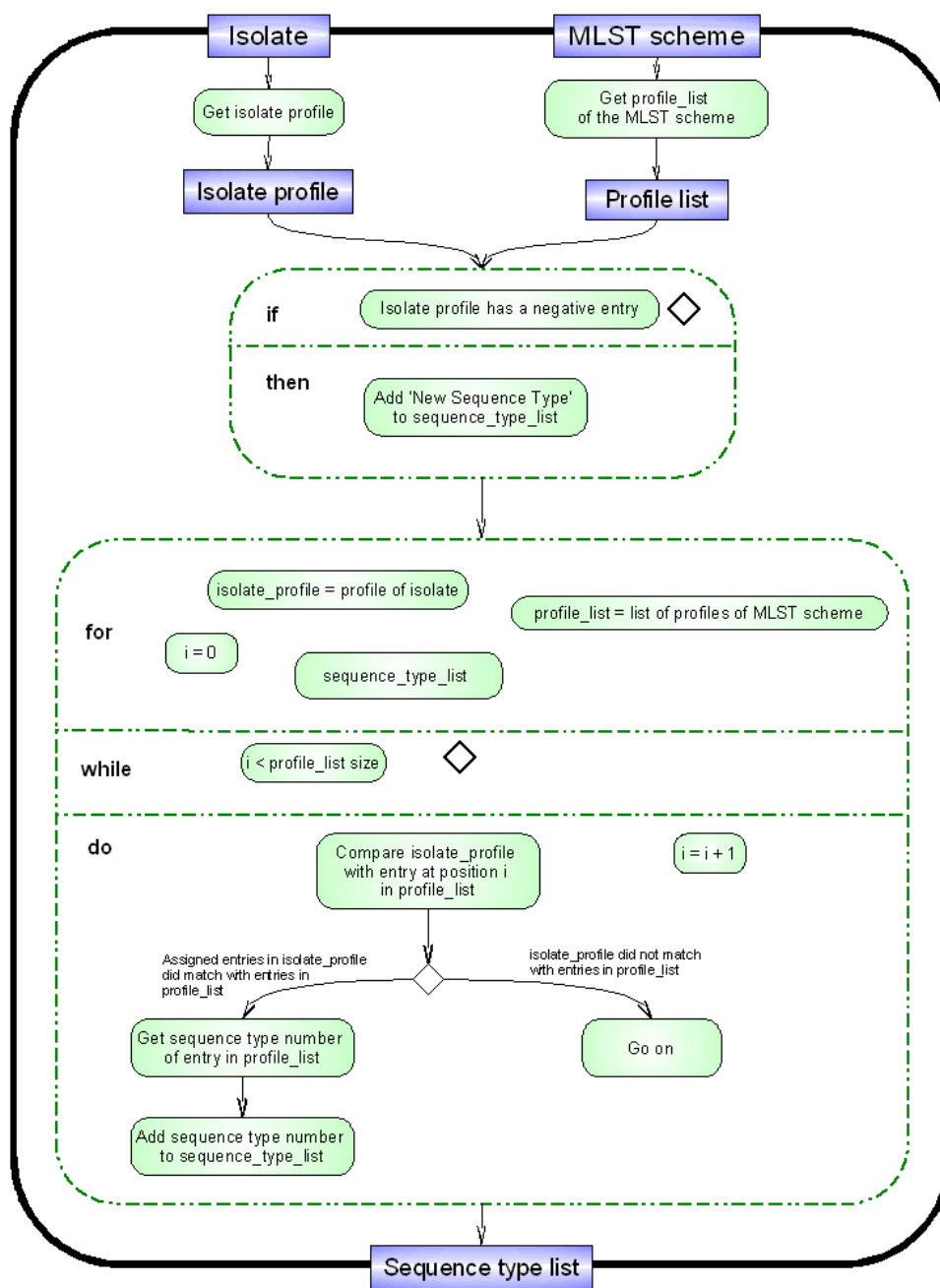
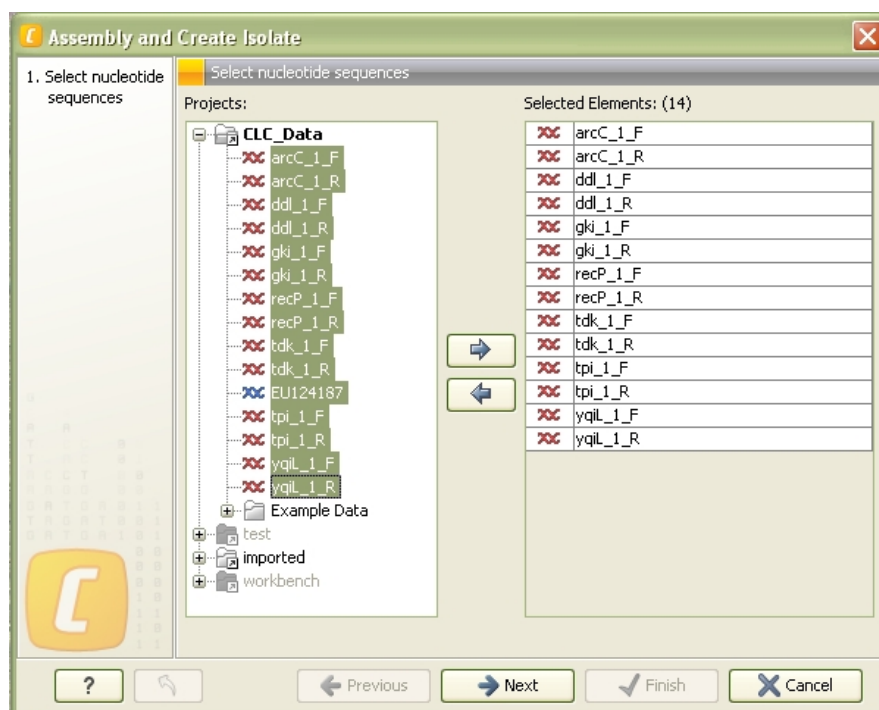


Figure 22: The activity diagram shows **the determination of a list of possible *sequence types*, the typing algorithm**. As soon as the *profile* of the *isolate* contains an entry with a negative number (this indicates a new *allele* type), the *sequence type* list will be extended with an entry of a 'New *sequence type*'. The *isolate profile* is compared to the entries of the *profile list* of the referenced *MLST scheme*. If the *profile* entries match, the *sequence type* of the *MLST scheme's profile* entry is determined and added to the *sequence type* list. Not assigned gene entries of the *isolate* will match with every entry, so the user gets a list of possible *sequence types*.

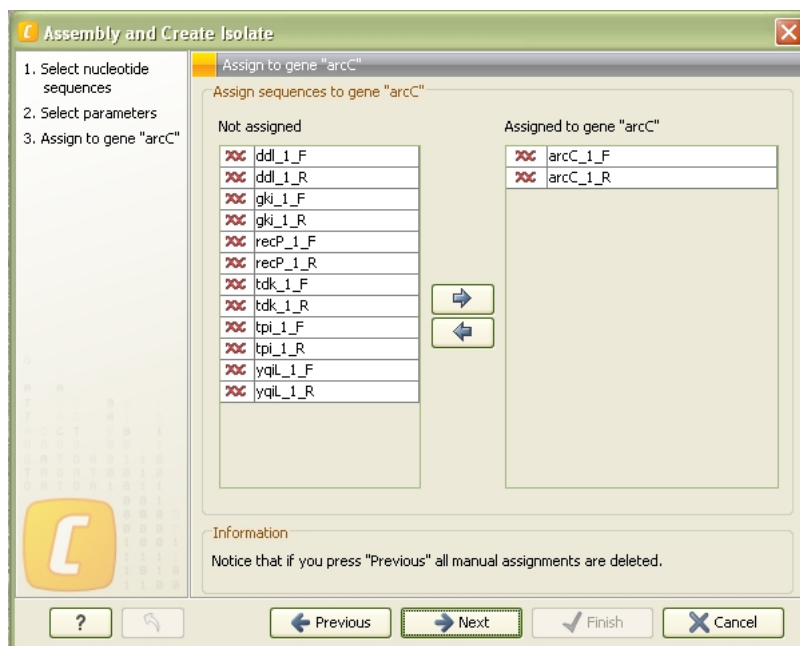


(a) Dialog window for selecting the input objects

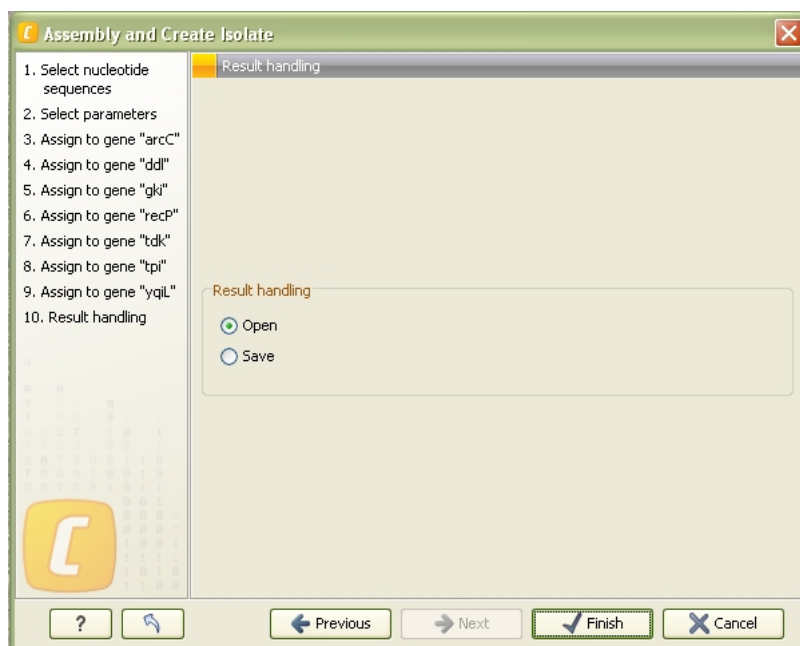


(b) Input parameter window

Figure 23: Dialog windows of the action “Assembly and Create Isolate” are shown by screenshots of the *CLC MLST Module 1.0*. The selection of the input objects (23a) is the first step. Afterwards, the *MLST* scheme, input parameters for the alignment and the trimming can be set (23b).

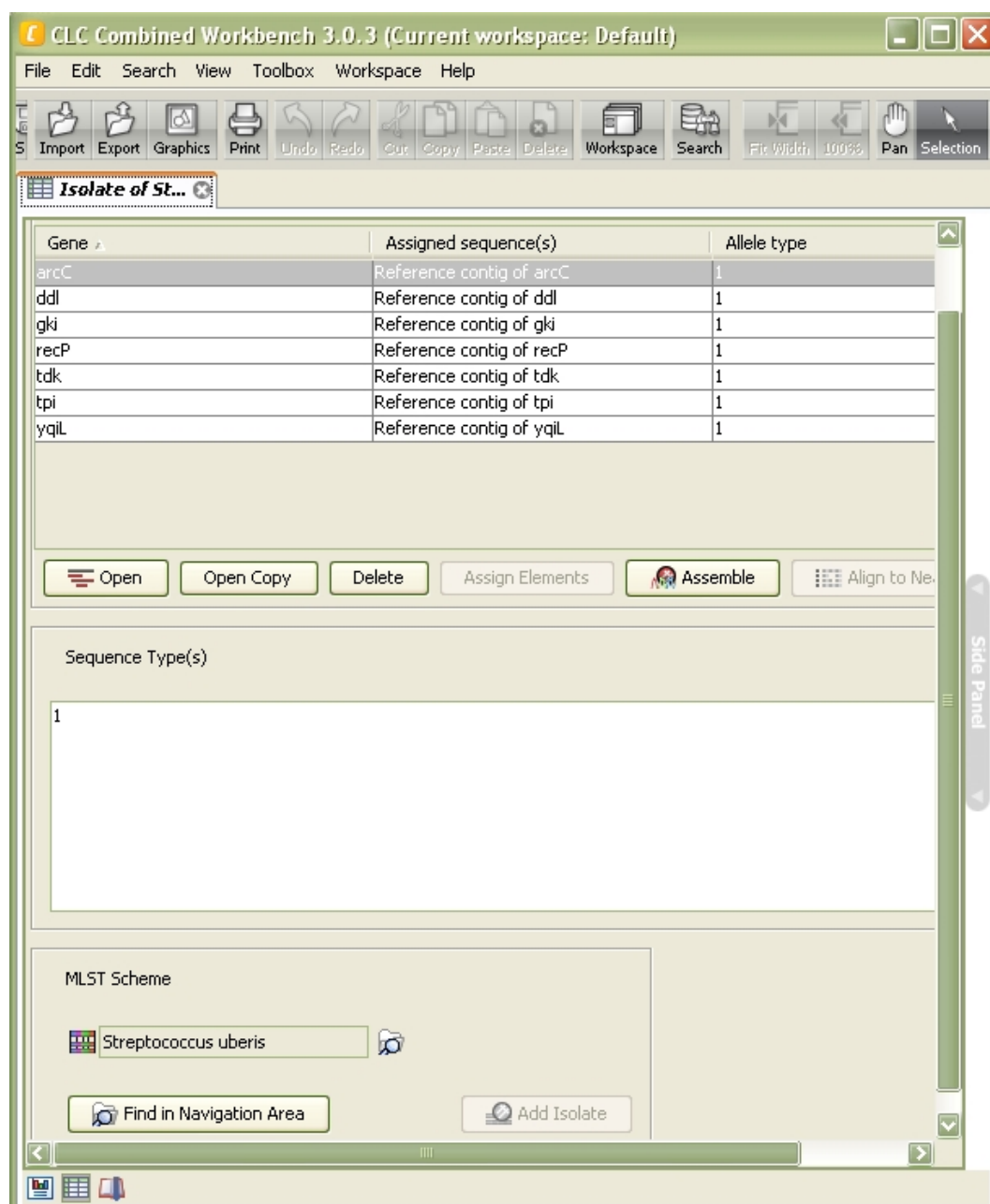


- (c) Automatic assignment for the gene “arcC” was performed in this step of the action. Two sequences could be assigned, the arcC_1_F and arcC_1_R sequence. The user is able to change the automatic assignment.



- (d) The window shows the options for handling the result. This step is similar to the result handling steps in all actions.

Figure 23: (Continued) Dialog windows of the action “Assembly and Create Isolate” are shown by screenshots of the *CLC MLST Module 1.0*. The Figure 23c reveals the automatic assignment of a gene. Users have the option of opening or saving the result immediately (Figure 23d).



(e) The result of the action “Assembly and Create Isolate” is a *Isolate* object. This is represented in this figure with one of the editor views.

Figure 23: (Continued) Dialog windows of the action “Assembly and Create Isolate” are shown by screenshots of the *CLC MLST Module 1.0*. The *Isolate* object comprises results and parameters of the action.

6.2.2 Create MLST scheme

Requirement 1 THE SYSTEM SHALL HAVE AN ACTION FOR CREATING NEW MLST SCHEMES

The action is called “Create MLST Scheme”.

Requirement 2 THE SYSTEM SHALL OFFER A FORM FOR USER INPUTS, THE GENE NAMES

The Figure 24a is a screenshot of the input window. Gene names and the name of the *MLST scheme* can be added. At least one gene name has to be added.

Requirement 3 THE SYSTEM SHALL OFFER A FORM FOR ADDING SEQUENCES

Users can select *allele* sequences with the creation of the *MLST scheme*. Those sequences are then added to the scheme (Figure 24b).

Requirement 4 IF THE USER ENTERS INVALID INPUT THE NEXT STEP CANNOT BE ENTERED OR THE ACTION CANNOT BE FINISHED

This is realized with the dialog windows of the action “Create *MLST scheme*”. The “Next” and “Finish” buttons (e.g. in Figure 24a) are enabled or disabled according to the entered parameters.

6.2.3 Download MLST scheme

Requirement 1 THE SYSTEM SHALL HAVE AN ACTION CONCERNING DOWNLOAD OF MLST SCHEMES

It is possible to download *MLST schemes* from PubMlst [1] with the *CLC MLST Module*. The name of the action is “Download *MLST schemes*”.

Requirement 2 THE SYSTEM DOWNLOADS THE NAMES OF THE AVAILABLE ORGANISMS

Figure 25 is a screenshot of the download action. This figure shows some of the organisms which are available from PubMlst [1] and MLST.net [19].

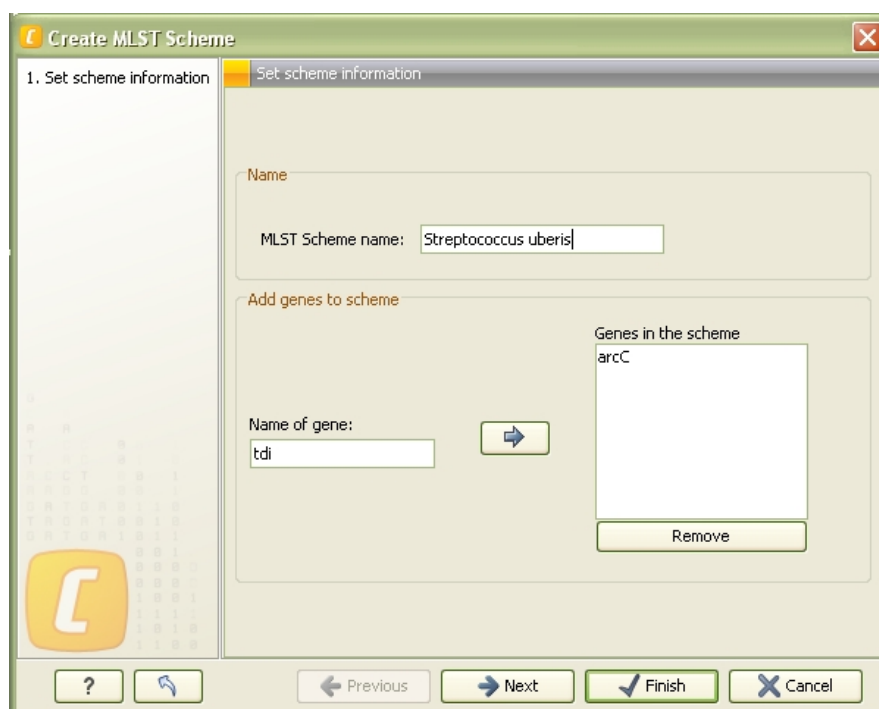
The names and links to the data of the available organisms are downloaded with an xml file from PubMlst [1], described in the Appendix A. This file is parsed and the organisms are represented, in order that the user can select the required *MLST schemes*. The links to the data files (profile file and fasta formatted sequence files) are saved temporarily.

Requirement 3 THE SYSTEM OFFERS A DIALOG TO SHOW THE ORGANISMS AND WHERE THE USER IS ABLE TO SET PARAMETERS (WHERE TO SAVE)

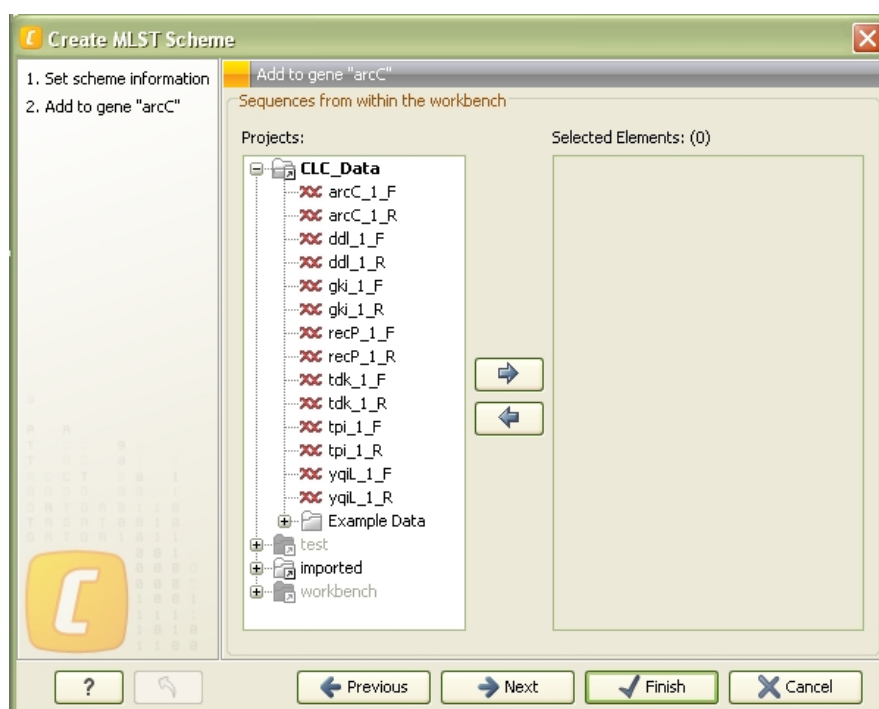
Figure 25 shows the organisms. Parameters for saving the resulting *MLST scheme* are set in a result handling window (compare the Figure 23d).

Requirement 4 THE SYSTEM SHALL DOWNLOAD MLST SCHEMES

The system downloads the *MLST schemes* with the information set by the user, i. e. the organisms. The links to the data files are downloaded with an



(a) Dialog window for the input of a new *MLST* scheme



(b) Input sequences can be added with the creation of a *MLST* scheme. These sequences are *allele* sequences of a specific gene (*arcC* in the example screenshot).

Figure 24: Dialog windows of the action “Create *MLST* scheme” are shown by screenshots of the *CLC MLST Module* 1.0.

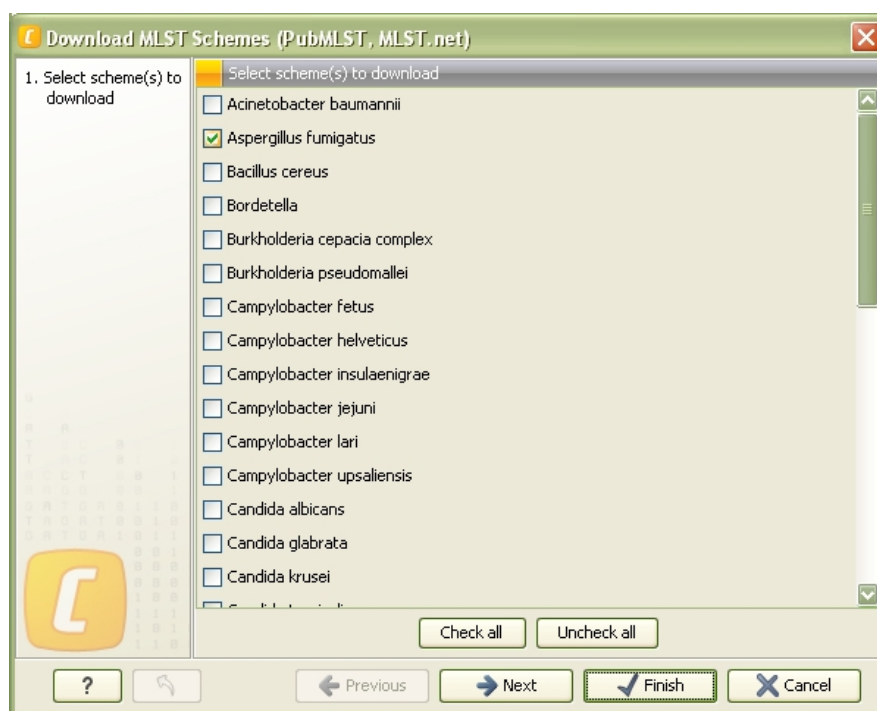


Figure 25: The download dialog window of the *CLC MLST Module* is shown in this figure. Some of the available organisms of PubMlst [1] are represented within the screenshot.

xml file and are now used to download the specific data files.

Requirement 5 THE SYSTEM SHALL SAVE MLST SCHEMES

The persistence of *MLST schemes* is handled with the *MLST scheme* object. The user can set an option in the dialog window of the action, whether the result object should be saved or just opened (compare Figure 23d, Page 46).

6.2.4 Merge MLST scheme

Requirement 1 THE SYSTEM SHALL HAVE AN ACTION FOR MERGING MLST SCHEMES

The name of the action is “Merge MLST Scheme” and is implemented.

Requirement 2 THE SYSTEM SHALL HAVE A DIALOG WHERE THE USER CAN SET THE MLST SCHEMES AND OTHER PARAMETERS

The action opens a dialog window. The user can set the *MLST schemes* and a parameter in this dialog window. The parameter handles the referencing of the *isolates*. Users can decide, whether the *isolates*, referenced to the input *MLST schemes*, should be referenced to the new merged *MLST scheme*.

Requirement 3 THE SYSTEM SHALL HAVE AN ALGORITHM MERGING TWO OR MORE MLST SCHEMES

The algorithm was developed. The entries in the first *MLST scheme* will be

kept in the new *MLST scheme*, whereas the other input *MLST schemes* are adapted to the first input *MLST scheme*'s entries. In fact, the first input *MLST scheme* is copied and the other input *MLST schemes* are adapted to the copy. The algorithm detects same entries with different *allele* numbers and adapts the entries.

Requirement 4 THE SYSTEM SHALL IDENTIFY NOT COMPATIBLE *MLST* SCHEMES
MLST schemes with different names or numbers of genes are not able to be merged. A different number of genes would cause an incomplete merged *MLST scheme*. Different names of the genes cause problems with the assignment of the *MLST schemes*'s genes. The algorithm is not merging incompatible *MLST schemes*.

6.2.5 Extend *MLST* scheme

Requirement 1 THE SYSTEM SHALL HAVE AN ACTION ASSOCIATED WITH EXTENDING *MLST* SCHEMES

The action "Add *Isolates* to *MLST scheme*" was implemented, in order to add *isolates*, representing new *sequence types*, to the referenced *MLST scheme*. In addition, an action called "Add Sequences to *MLST scheme*" was implemented to add allele sequences to the genes of an *MLST scheme*.

Requirement 2 THE SYSTEM SHALL HAVE A DIALOG WINDOW, WHERE THE USER CAN SELECT THE ISOLATES AND SET PARAMETERS

A dialog window was developed for the user to select the *isolates*, intended for the addition to its referenced *MLST scheme*. Furthermore, the dialog window for adding sequences to the *MLST scheme* was developed, which is also used in the action "Create *MLST scheme*" (Section 6.2.2 and represented in the Figure 24b, Page 49).

Requirement 3 THE SYSTEM SHALL HAVE AN ALGORITHM, WHICH ADDS ISOLATES AS SEQUENCE TYPES TO AN *MLST* SCHEME

The algorithm is implemented for adding *isolates* to their referenced *MLST schemes*. *Isolates* with one negative number in the array for the possible *sequence types* are valid for addition to the *MLST scheme*, since this indicates that this *isolate* is a new *sequence type*. First of all the sequences or consensus sequences of *contigs* are added to the *MLST scheme*. Then the *Isolate* object is updated with the extended *MLST scheme*. In the end the updated profile (*allele* numbers) of the *isolate* is added to the profiles of the *MLST scheme*.

Requirement 4 THE SYSTEM SHALL DISTINGUISH BETWEEN NEW AND DEFINED SEQUENCE TYPES AS ISOLATES

The *Isolate* object contains an array of integers, representing the possible *sequence types*. An *isolate* is defined, if the array is of length one and the entry is not negative. A negative entry indicates a possible new *sequence type*. Only *Isolates* with an array length of one and a negative entry can be added

to a *MLST scheme*, since this indicates a new *sequence type*.

Requirement 5 THE SYSTEM SHALL HAVE AN UPDATE FUNCTIONALITY, WHICH UPDATES AN ISOLATE WITH RESPECT TO ITS ASSOCIATED MLST SCHEME. UPDATE AN ISOLATE MEANS TO TYPE THE ISOLATE. Typing was implemented (Section 6.2.1, Page 40).

6.2.6 Extend Isolate

Requirement 1 THE SYSTEM SHALL HAVE AN ACTION ASSOCIATED WITH THE EXTENSION OF ISOLATES
The action is called “Assemble to Existing *Isolate*” and was implemented reusing the already finished code of the action “Assembly and Create *Isolate*”.

Requirement 2 THE SYSTEM OFFERS AN INPUT FORM FOR PARAMETERS AND A DIALOG WINDOW
The input objects are the *isolate* and sequences, which shall be added. Those objects are selected in the first step of the dialog window for the “Assemble to Existing *Isolate*” action. Afterwards parameters for the alignment and trimming can be set. The user can finish or continue to the next step (only if parameters are valid) and the automatic assigning follows, see Figure 20 at Page 42. This action is not implemented completely new, it is derived of the *typing* workflow implemented in the action “Assembly and Create *Isolate*”. The result is not a newly created *Isolate* object, the result information is added to the *Isolate* object, which was selected at the beginning of the action.

Requirement 3 THE SYSTEM HAS AN ALGORITHM FOR TYPING
Typing was implemented, see Section 6.2.1 at Page 40.

Requirement 4 THE SYSTEM SHALL ADD INFORMATION TO EXISTING ISOLATE OBJECTS
The *Isolate* object contains 1 - 11 objects which can be sequence objects or *contig* objects. Addition of sequences to an *Isolate* object is handled with the following description.

- If the *isolate* doesn't contain an element associated with a gene, the created object (*contig* or sequence) is added to the *Isolate* object.
- If there has already been assigned a sequence object to a gene and the algorithm tries to add another sequence object, all sequences are assembled into a *contig*.
- The addition of a sequence to a *contig* is performed with the existing algorithm “Add Sequences to Contig” of the CLC Workbench. The sequence can affect the consensus sequence of the *contig*.

6.2.7 Submission of data

The feature “Submission of data” is the counterpart of the system feature “Download MLST scheme”. The download feature downloads *MLST schemes* from PubMlst.org [1], as described in Section 6.2.3 at Page 48. The submission feature should upload new acquired data from typing to a public database.

The submission of data is handled via emails. There are specific sheets, which have to be filled out by the submitter. The forms have to be attached to an email and have to be sent to a curator of the specific organism. This curator decides whether the data is valid and will be added to the database or the data will be ignored. Since there is a reorganization of the submission function on the database side of PubMlst.org [1], the *CLC MLST Module* will not include the system feature “Submission of data”. This feature is postponed to a later version of the software.

6.3 Additional features

6.3.1 Automatic update

In order to ease the handling of all data regarding the molecular typing with *multilocus sequence typing*, an automatic update functionality was introduced. The update functionality uses the developed algorithm of typing in Section 6.2.1 at Page 40.

The following situations show the benefit of an automatic update.

Problem 1

The user changes the *MLST scheme*, for example he/she deletes a sequence or a profile. All *isolates* have to be updated again, also *isolates*, which have been typed long time ago.

Problem 2

The isolate has to be changed since for example the *contig* has a wrongly identified nucleotide. The isolate has to be updated again.

Solution

The stated problems include repeated updates/typing of the *isolates*. This is very time-consuming and unnecessary.

The solution for problem 1 is a version concept for the *MLST scheme*. The *MLST scheme* has a version. On every change of the object the version also changes. The *isolate* contains a reference of the *MLST scheme* and the version, which was used to update the last time. If the user opens the isolate in a view, the version is checked and if the *isolate*’s version differs to the version in the *MLST scheme*, a dialog window, see in Figure 26, is shown. The user has the possibility of updating the *Isolate* object or keeping the object in the state it is. If the user cancels the dialog

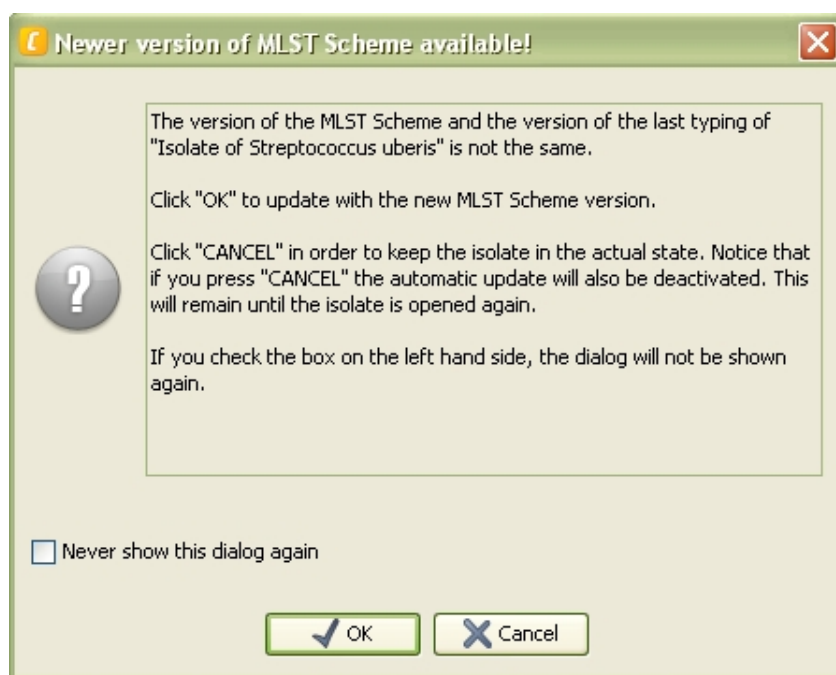


Figure 26: The automatic update dialog window is activated with opening an *isolate* and differing versions of the *isolate* and its referenced *MLST scheme*.

window, the *isolate* state is kept and the automatic update will not be performed until the *isolate* is opened again.

The solution for problem 2 is a listener which listens to every change in the *isolate*. The typing algorithm is performed when a change happens.

6.3.2 Popup menus in views

The views of the objects, described in Section 6.4, have popup menus with the right mouse click. These menus provide fast workflows of data extraction.

For example, the *MLST scheme profiles* table view (6.4.3) has a popup menu for extracting joined sequences, the concatenated sequence of a *sequence type* as well as removing *sequence types*. In addition, it is possible to align *sequence types* with this popup menu.

The *MLST scheme allele* table view (6.4.3) has a popup menu for extracting allele sequences into a sequence list object or in an alignment. Furthermore, with this popup menu, allele sequences can be removed.

6.3.3 History

With every object in the CLC Workbenches, a history is attached. The history collects all changes, performed on an object. The utility of a history messaging system is available with the Developer Kit and was implemented in the *CLC MLST*

Module. All changes are recorded with parameters, so that the user has an overview of all tasks according to an object. The Figure 27 is the history view of an *MLST scheme*.

6.3.4 Undo/Redo

The Developer Kit provides a framework for undo and redo operations on objects. This framework was used to implement the undo/redo functionality.

6.4 Graphical user interface

6.4.1 History view

The history of an object in a CLC Workbench (Section 6.3.3) is a log file for recording all changes with parameters of this object. Thus, the user has an overview of all changes in the object. Figure 27 shows some changes in an *MLST scheme* object.

6.4.2 Isolate editors and views

Isolate table view

The main view of an *isolate* is the *Isolate table view*. This view shows the assigned objects in the table. The table displays the gene, its assigned object and the allele type, resulted from the typing algorithm. In Figure 28, the gene “arcC” and “ddl” were identified as new *allele types*, whereas the other *allele* sequences were identified as *allele type* one. Below the table are buttons for opening, deleting and changing the assigned objects. The “Align to Nearest” button aligns the assigned sequence (only possible with a new *allele type*) to the *allele* sequences of the *MLST scheme* with the alignment functionality of the CLC Workbench, in order to find the nearest *allele* sequence. In the middle of the view, the possible *sequence types* are shown. Figure 28 represents a possible new *sequence type*. At the bottom is information of the referenced *MLST scheme*. Three buttons were implemented for finding the referenced *MLST scheme* in the navigator (displayed on the left hand side in the CLC Workbench), changing the referenced *MLST scheme* and for adding this *isolate* to its *MLST scheme*.

Isolate report view

This view collects all information regarding the typing of the isolate. The result is presented in a printable layout for the user.

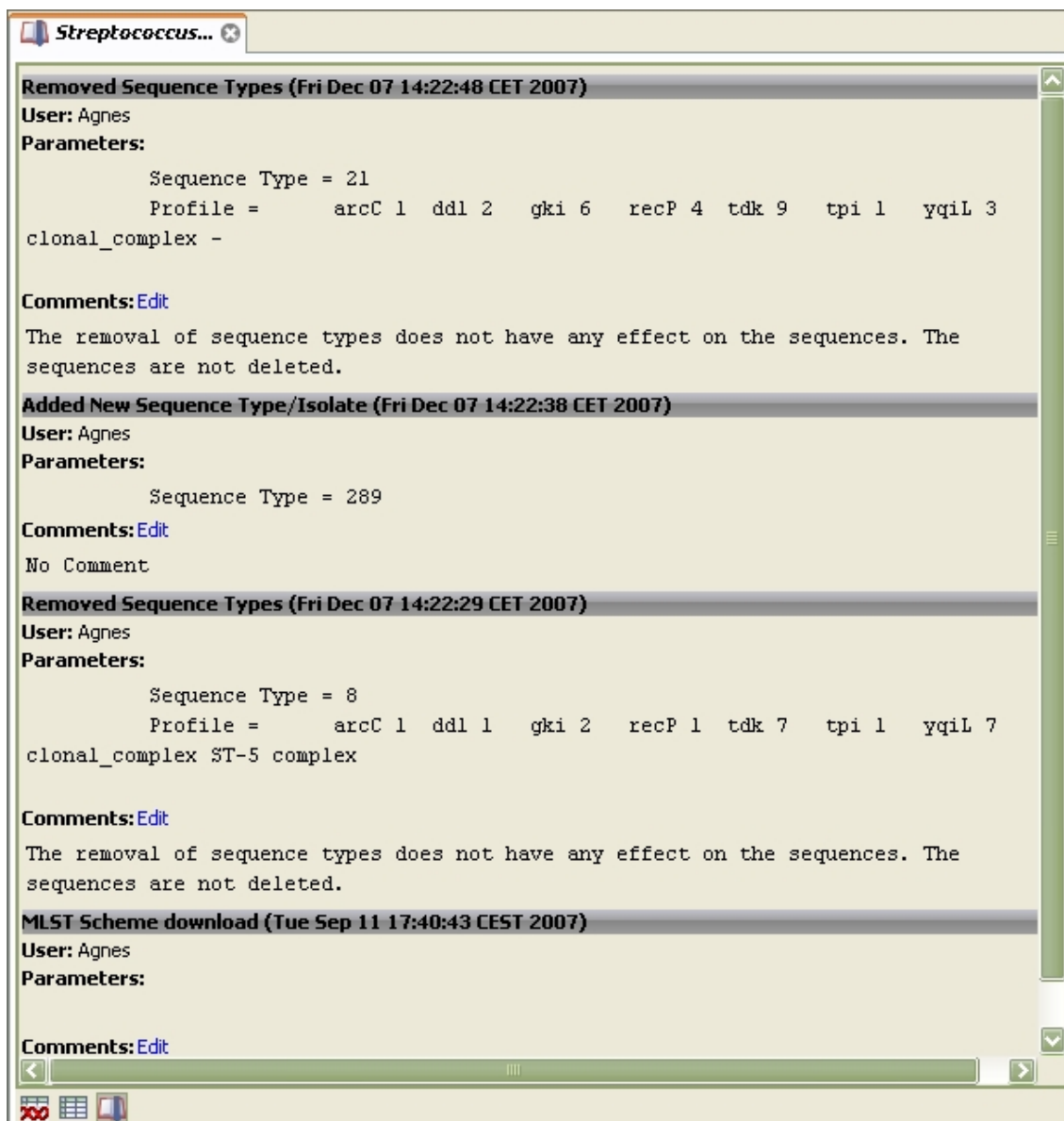


Figure 27: History entries are shown in a history view in every object in the CLC Workbench. This is a *MLST scheme* representing some changes.

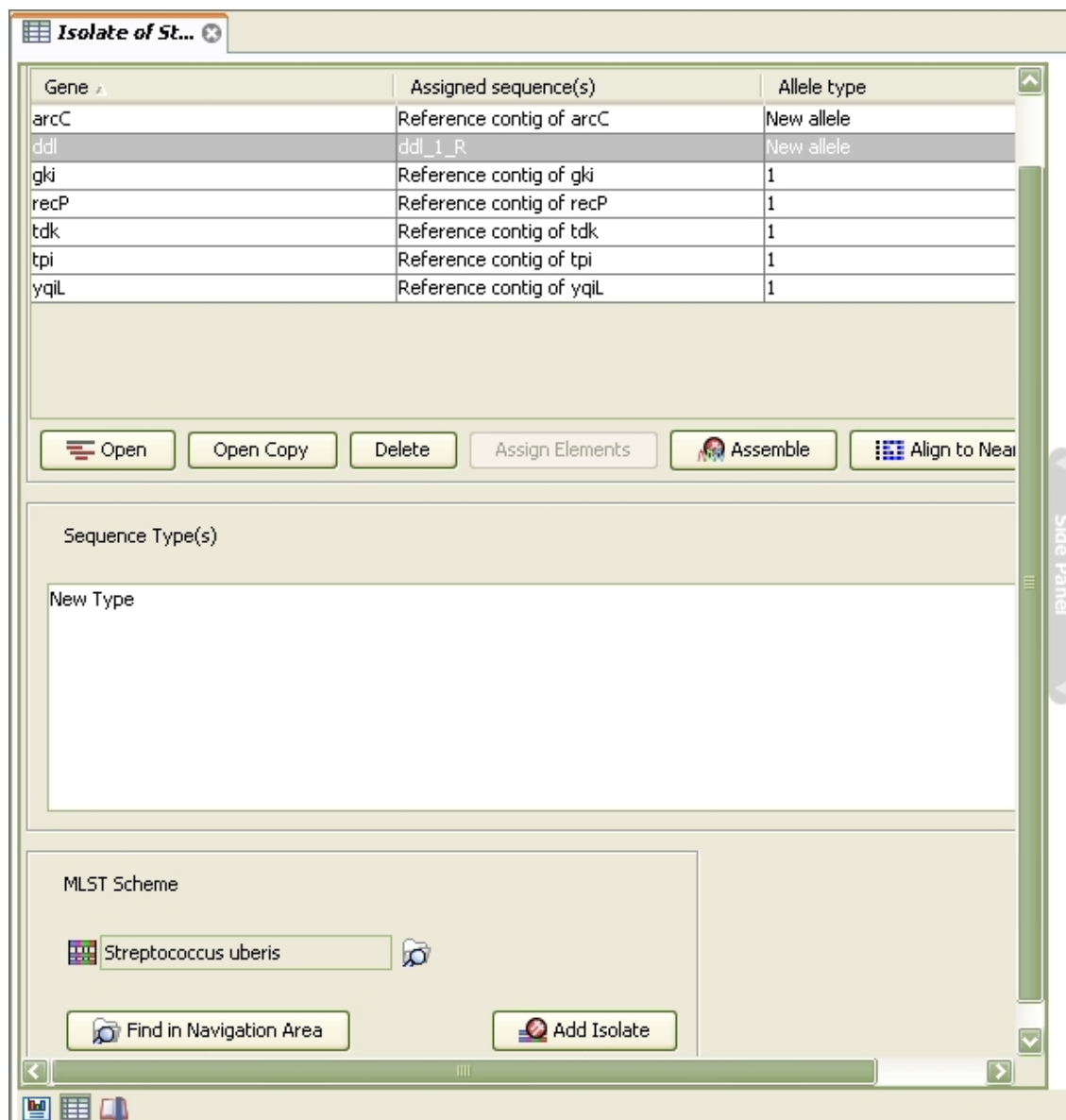


Figure 28: The main view of the *Isolate* object. All assigned objects are shown for the genes as well as the referenced *MLST* scheme.

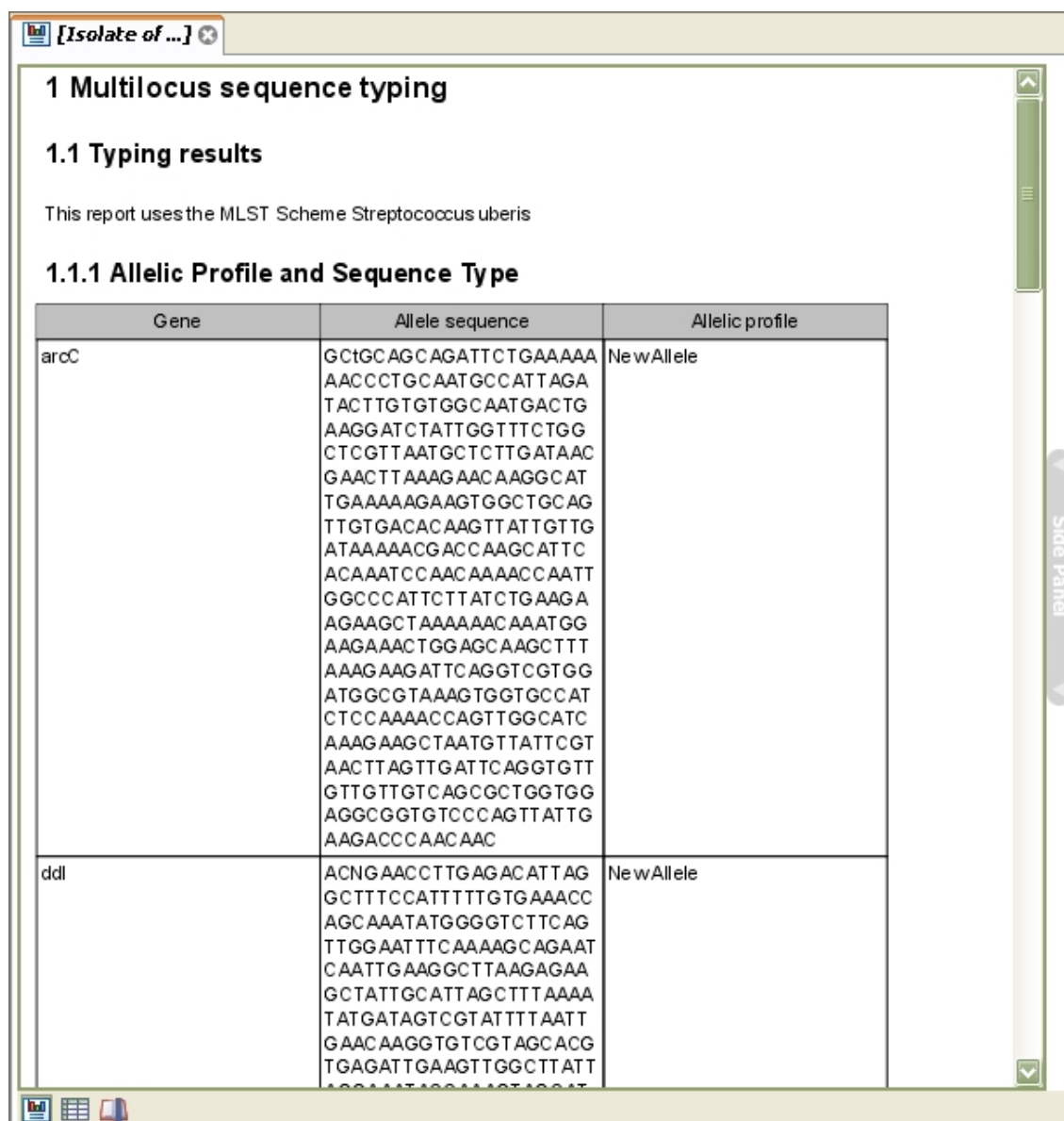


Figure 29: This figure represents the view of all collected informations of the typing for this *isolate*, the *Isolate* report view.

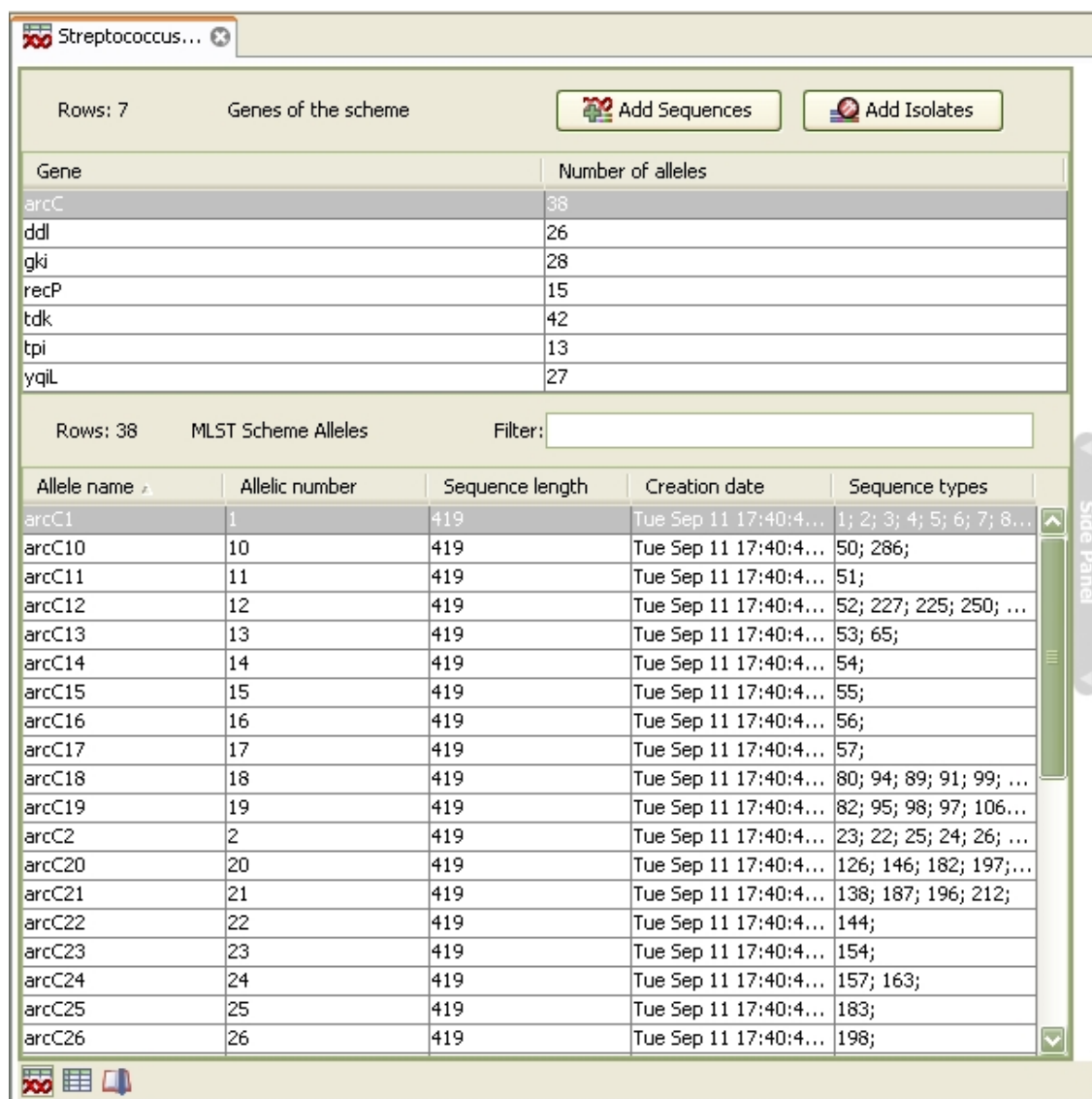
6.4.3 MLST scheme editors and views

Allele table view

This view contains all information about the alleles represented in the *MLST scheme*, see in Figure 30. The first table contains the genes and the number of alleles. The user selects one row and the second table shows the sequences contained in the selected gene with specific information. Two buttons are available on the upper side. The “Add Sequences” button opens an action for adding sequences to the genes of the *MLST scheme*. The “Add Isolates” button opens a dialog window, containing the referenced *isolates*, standing for new *sequence types*. There, the user is able to select *isolates* for the addition as *sequence types* to the *MLST scheme*.

Profiles table view

The *Profiles* table view, Figure 31, displays all included profiles of a *MLST scheme*. With the filter option on the upper side the user can refine the shown profiles.



The screenshot shows a web application window titled "Streptococcus...". It contains two main sections. The top section, "Genes of the scheme", shows a table with 7 rows of genes and their allele counts. The bottom section, "MLST Scheme Alleles", shows a table with 38 rows of allele details, including name, number, length, creation date, and sequence types. A "Filter:" input field is present above the alleles table. A "Side Panel" is visible on the right side of the alleles table.

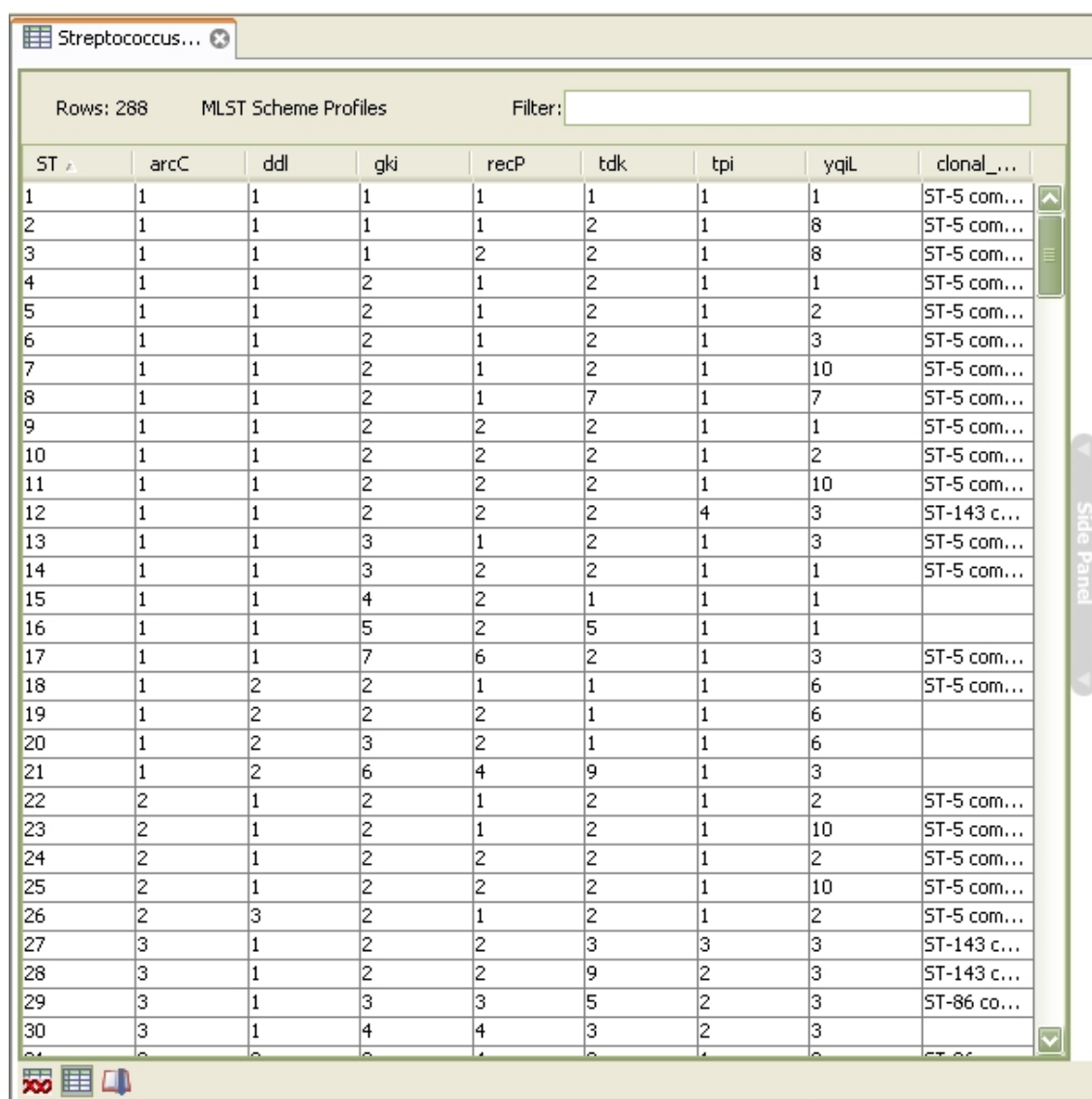
Rows: 7 Genes of the scheme Add Sequences Add Isolates

Gene	Number of alleles
arcC	38
ddl	26
gki	28
recP	15
tdk	42
tpi	13
yqil	27

Rows: 38 MLST Scheme Alleles Filter:

Allele name	Allelic number	Sequence length	Creation date	Sequence types
arcC1	1	419	Tue Sep 11 17:40:4...	1; 2; 3; 4; 5; 6; 7; 8...
arcC10	10	419	Tue Sep 11 17:40:4...	50; 286;
arcC11	11	419	Tue Sep 11 17:40:4...	51;
arcC12	12	419	Tue Sep 11 17:40:4...	52; 227; 225; 250; ...
arcC13	13	419	Tue Sep 11 17:40:4...	53; 65;
arcC14	14	419	Tue Sep 11 17:40:4...	54;
arcC15	15	419	Tue Sep 11 17:40:4...	55;
arcC16	16	419	Tue Sep 11 17:40:4...	56;
arcC17	17	419	Tue Sep 11 17:40:4...	57;
arcC18	18	419	Tue Sep 11 17:40:4...	80; 94; 89; 91; 99; ...
arcC19	19	419	Tue Sep 11 17:40:4...	82; 95; 98; 97; 106...
arcC2	2	419	Tue Sep 11 17:40:4...	23; 22; 25; 24; 26; ...
arcC20	20	419	Tue Sep 11 17:40:4...	126; 146; 182; 197;...
arcC21	21	419	Tue Sep 11 17:40:4...	138; 187; 196; 212;
arcC22	22	419	Tue Sep 11 17:40:4...	144;
arcC23	23	419	Tue Sep 11 17:40:4...	154;
arcC24	24	419	Tue Sep 11 17:40:4...	157; 163;
arcC25	25	419	Tue Sep 11 17:40:4...	183;
arcC26	26	419	Tue Sep 11 17:40:4...	198;

Figure 30: This figure represents the view of a *MLST scheme* object with its allele sequences in tables.



Streptococcus... Rows: 288 MLST Scheme Profiles Filter:

ST	arcC	ddl	gki	recP	tdk	tpi	yqil	clonal_...
1	1	1	1	1	1	1	1	ST-5 com...
2	1	1	1	1	2	1	8	ST-5 com...
3	1	1	1	2	2	1	8	ST-5 com...
4	1	1	2	1	2	1	1	ST-5 com...
5	1	1	2	1	2	1	2	ST-5 com...
6	1	1	2	1	2	1	3	ST-5 com...
7	1	1	2	1	2	1	10	ST-5 com...
8	1	1	2	1	7	1	7	ST-5 com...
9	1	1	2	2	2	1	1	ST-5 com...
10	1	1	2	2	2	1	2	ST-5 com...
11	1	1	2	2	2	1	10	ST-5 com...
12	1	1	2	2	2	4	3	ST-143 c...
13	1	1	3	1	2	1	3	ST-5 com...
14	1	1	3	2	2	1	1	ST-5 com...
15	1	1	4	2	1	1	1	
16	1	1	5	2	5	1	1	
17	1	1	7	6	2	1	3	ST-5 com...
18	1	2	2	1	1	1	6	ST-5 com...
19	1	2	2	2	1	1	6	
20	1	2	3	2	1	1	6	
21	1	2	6	4	9	1	3	
22	2	1	2	1	2	1	2	ST-5 com...
23	2	1	2	1	2	1	10	ST-5 com...
24	2	1	2	2	2	1	2	ST-5 com...
25	2	1	2	2	2	1	10	ST-5 com...
26	2	3	2	1	2	1	2	ST-5 com...
27	3	1	2	2	3	3	3	ST-143 c...
28	3	1	2	2	9	2	3	ST-143 c...
29	3	1	3	3	5	2	3	ST-86 co...
30	3	1	4	4	3	2	3	

Figure 31: This view of a *MLST* scheme shows the profiles contained in the object.

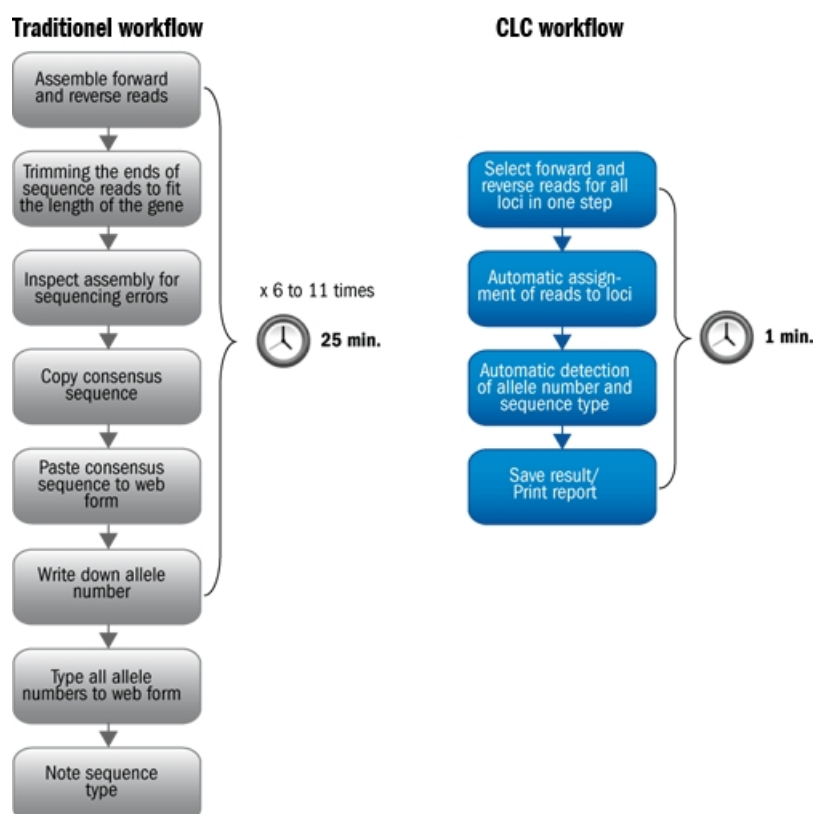


Figure 32: MLst workflow. This figure shows the difference of time spent for typing an *isolate* with the *CLC MLST Module* and an online form, e.g. PubMlst [1].

7 Results

The result of this thesis is a software module, the *CLC MLST Module*. This module is a plug-in for the CLC bio Workbench (DNA and Combined) and is particularly suitable for rapid molecular typing with *Multilocus sequence typing* and the management of *MLST* related data.

7.1 Overall evaluation

The module provides a fast workflow for typing with *MLST*. Figure 32 compares the expenditure of time between an online form, e.g. PubMlst [1] or MLST.net [19], and the *CLC MLST Module*. It shows that the *CLC MLST Module* can bring down the time, spent on one isolate, to one minute. However, one must know that the potential editing of the *contig* can extend the amount of time. Based on the quality of the sequence reads, there could be the need of reviewing the *contig*.

Contrary to other software solutions, see Section 4.3, the module offers a flexible environment. The dynamic update functionality highly enhances the handling of the data.

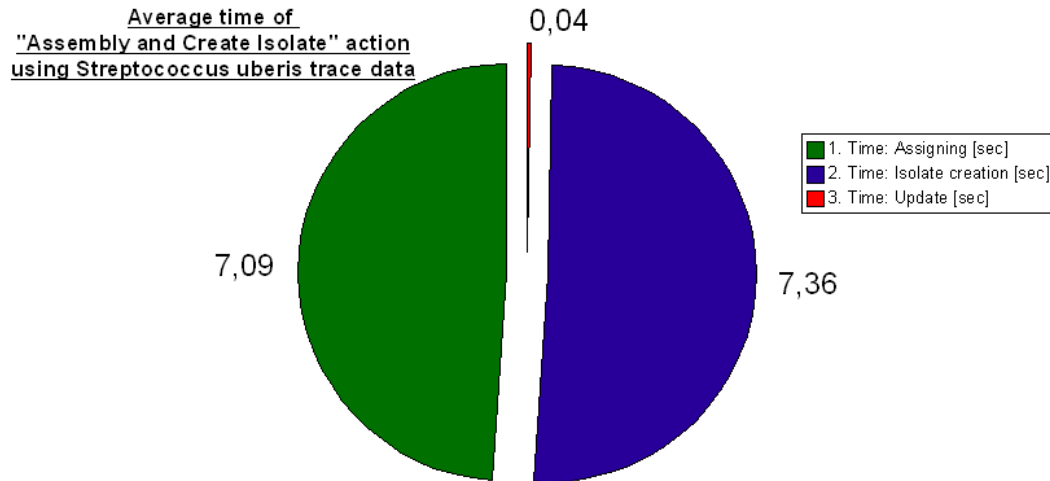


Figure 33: Evaluation of *trace* files, provided by PubMlst [1], for organism *Streptococcus uberis*.

7.2 Detailed evaluation

For evaluation purposes, available reads of typed isolates from PubMlst [1] were downloaded. This data was used to test the *MLST* framework of the *CLC MLST Module*.

Trace data, the fluorescent peak traces of sequenced *DNA*, of the organism *Streptococcus uberis* was used for analysing the time expenditure of the algorithm. Altogether, 86 *trace* files were downloaded. Those files construct 118 *sequence types*. The algorithm “Assemble and Create Isolate”, the *typing* workflow, is used and can be separated into three steps:

1. Arrange *trace* data to genes
2. Create *isolate* (assemble the *trace* data)
3. Update *isolate* with its referenced *MLST scheme*

In this evaluation, 110 *isolates* were typed correctly with the workflow. The average time, spent on the three steps, are illustrated in Figure 33 (processor with 1,6 GHz, 504 MB RAM). Eight constructed *isolates* were identified as new *sequence types*. This comes from inconsistencies in the *contigs*. In conclusion, the inspection of the resulted *contigs* by the user is part of the workflow and cannot be automated. The user can use low quality reads, then, the expenditure of time will increase, since it causes more manual inspections.

The result of the the evaluation with the *trace* data shows that the amount of time, spent on the update, is below the non-functional performance requirement of 1 second.

7.3 User acceptance

The concept of the CLC MLST Module and a beta version of the software were presented to possible customers on the “17th European Congress of Clinical Microbiology and Infectious Diseases” and the “25th International Congress of Chemotherapy” in Munich. The software was well received.

8 Discussion

The aim of this project was to develop a plug-in for the software solution of CLC bio [6], performing *multilocus sequence typing*. The result is the *CLC MLST Module*, which is a user-friendly solution for rapid typing with *multilocus sequence typing* and managing the acquired data.

The following items describe future prospects, pointing out the subsequent steps for enhancing and improving the *CLC MLST Module*.

Performance

Performance reduction, caused by memory leaks, are tried to be eliminated. Profiling was done and the performance increased. However, it seems that the software's performance is not reached, yet. Therefore one task for the future is another profiling of the *CLC MLST Module*.

More automation

The system could have more automation for rapid typing. The workflow enhances the typing, however, the user has to manually select the sequences for each *isolate*. If the user saves the sequences in a defined order, the system could import and assign the sequences itself.

Multilocus sequence analysis

In Figure 2, the phases of an *MLST* project are described. The preparation and data collection is handled in laboratories. The second phase, the *multilocus sequence typing* can be achieved by using the *CLC MLST Module*. The last phase, the *multilocus sequence analysis*, can be performed with functionality of the CLC Workbench. However, useful clustering methods like eBURST [28, 29, 30] and BURP [31] are not implemented for the *CLC MLST Module*, yet.

Download from more databases

The download functionality is restricted to the download from PubMlst [1] for now. The implementation of this download is developed with an abstract concept, in order to facilitate the import of other databases, like the SpaServer [11]. Next development steps could include the download of more databases.

Submission

The feature of submission of data to public databases was not implemented. At the time of this thesis, the submission for the PubMlst [1] database was restructured. The user is able to submit data by using the online form or email. Thus, the

feature was postponed. In order to fulfill the functionality of the entire workflow of *multilocus sequence typing*, the easy submission of data should be implemented.

SOAP

SIMPLE OBJECT ACCESS PROTOCOL (SOAP) is a protocol, intended for exchanging information in a distributed system using XML technologies. At the time of this thesis, Keith A. Jolley used this protocol for building an API (Application Programming Interface) for the PubMlst [1] database. This API provides functions for querying the MLST databases at PubMlst. For future prospects this functions could be included into the *CLC MLST Module* software. There are functions available for typing sequences, which could substitute the typing functionality of the *CLC MLST Module*. However, the *CLC MLST Module* gives the opportunity of changing, extending and managing *MLST* related data as well as being faster than a protocol-based typing procedure. Time plays a major role in the dynamic updates of the *isolates*. Therefore the SOAP API can be an extension in the *CLC MLST Module*, but will not replace the developed functionalities.

9 Conclusion

The *CLC MLST Module* is a versatile solution for *Multilocus sequence typing* with focus on usability, the typing workflow and maintenance of *MLST* data. The rapid typing with *MLST* is improved and the managing of *MLST* related data is facilitated.

The first version of the *CLC MLST Module* was released (*CLC MLST Module* 1.0 [7]) and it indicates to be an efficient and stable software solution for *multilocus sequence typing*.

10 References

- [1] Keith A. Jolley. PubMlst website - Publicly-accessible MLST databases. <http://pubmlst.org>.
- [2] M. C. Maiden, J. A. Bygraves, E. Feil, G. Morelli, J. E. Russell, R. Urwin, Q. Zhang, J. Zhou, K. Zurth, D. A. Caugant, I. M. Feavers, M. Achtman, and B. G. Spratt. Multilocus sequence typing: a portable approach to the identification of clones within populations of pathogenic microorganisms. *Proc Natl Acad Sci U S A*, 95(6):3140–3145, Mar 1998.
- [3] D. M. Hacek, T. Suriano, G. A. Noskin, J. Kruszynski, B. Reisberg, and L. R. Peterson. Medical and economic benefit of a comprehensive infection control program that includes routine determination of microbial clonality. *Am J Clin Pathol*, 111(5):647–654, May 1999.
- [4] Aparajita Singh, Richard V Goering, Shabbir Simjee, Steven L Foley, and Marcus J Zervos. Application of molecular techniques to the study of hospital infection. *Clin Microbiol Rev*, 19(3):512–530, Jul 2006.
- [5] George M. Garrity, editor. *Bergey’s Manual of Systematic Bacteriology*. Springer-Verlag, second edition, 2005.
- [6] CLC bio - Bioinformatics solutions. <http://www.clcbio.com/>. CLC bio provides software solutions for bioinformatics.
- [7] CLC bio - CLC MLST Module. www.clcbio.com/mlst. The CLC MLST Module is a plug-in for performing multilocus sequence typing.
- [8] John W Taylor and Matthew C Fisher. Fungal multilocus sequence typing—it’s not just for bacteria. *Curr Opin Microbiol*, 6(4):351–356, Aug 2003.
- [9] Phaedra Agius, Barry Kreiswirth, Steve Naidich, and Kristin Bennett. Typing staphylococcus aureus using the spa gene and novel distance measures. *IEEE/ACM Trans Comput Biol Bioinform*, 4(4):693–704, 2007.
- [10] H. M. Frénay, A. E. Bunschoten, L. M. Schouls, W. J. van Leeuwen, C. M. Vandenbroucke-Grauls, J. Verhoef, and F. R. Mooi. Molecular typing of methicillin-resistant staphylococcus aureus on the basis of protein a gene polymorphism. *Eur J Clin Microbiol Infect Dis*, 15(1):60–64, Jan 1996.
- [11] SpaServer. <http://www.seqnet.org/>. SpaServer contains data for spa-typing of Staphylococcus aureus.
- [12] Tamara Revazishvili, Mamuka Kotetishvili, O. Colin Stine, Arnold S Kreger, J. Glenn Morris, and Alexander Sulakvelidze. Comparative analysis of multilocus sequence typing and pulsed-field gel electrophoresis for characterizing listeria monocytogenes strains isolated from environmental and clinical sources. *J Clin Microbiol*, 42(1):276–285, Jan 2004.

- [13] Mamuka Kotetishvili, O. Colin Stine, Yuansha Chen, Arnold Kreger, Alexander Sulakvelidze, Shanmuga Sozhamannan, and J. Glenn Morris. Multilocus sequence typing has better discriminatory ability for typing vibrio cholerae than does pulsed-field gel electrophoresis and provides a measure of phylogenetic relatedness. *J Clin Microbiol*, 41(5):2191–2196, May 2003.
- [14] Mamuka Kotetishvili, O. Colin Stine, Arnold Kreger, J. Glenn Morris, and Alexander Sulakvelidze. Multilocus sequence typing for characterization of clinical and environmental salmonella strains. *J Clin Microbiol*, 40(5):1626–1635, May 2002.
- [15] Crystal N Johnson, William H Benjamin Jr, Stephen A Moser, Susan K Hollingshead, Xiaotian Zheng, Marilyn J Crain, Moon H Nahm, and Ken B Waites. Genetic relatedness of levofloxacin-nonsusceptible streptococcus pneumoniae isolates from north america. *J Clin Microbiol*, 41(6):2458–2464, Jun 2003.
- [16] S. J. Peacock, G. D I de Silva, A. Justice, A. Cowland, C. E. Moore, C. G. Winearls, and N. P J Day. Comparison of multilocus sequence typing and pulsed-field gel electrophoresis as tools for typing staphylococcus aureus isolates in a microepidemiological setting. *J Clin Microbiol*, 40(10):3764–3770, Oct 2002.
- [17] Sreedhar R Nallapareddy, Ruay-Wang Duh, Kavindra V Singh, and Barbara E Murray. Molecular typing of selected enterococcus faecalis isolates: pilot study using multilocus sequence typing and pulsed-field gel electrophoresis. *J Clin Microbiol*, 40(3):868–876, Mar 2002.
- [18] S. Sreevatsan, X. Pan, K. E. Stockbauer, N. D. Connell, B. N. Kreiswirth, T. S. Whittam, and J. M. Musser. Restricted structural gene polymorphism in the mycobacterium tuberculosis complex indicates evolutionarily recent global dissemination. *Proc Natl Acad Sci U S A*, 94(18):9869–9874, Sep 1997.
- [19] David Aanensen. MLST.net website. www.mlst.net.
- [20] Mark Achtman. MLST Databases at the Max Planck Institute for Infection Biology. <http://web.mpiib-berlin.mpg.de/mlst/>.
- [21] Keith A Jolley, Man-Suen Chan, and Martin C J Maiden. mlstdbnet - distributed multi-locus sequence typing (mlst) databases. *BMC Bioinformatics*, 5:86, Jul 2004.
- [22] The Staden package. <http://staden.sourceforge.net/>.
- [23] R. Staden. The staden sequence analysis package. *Mol Biotechnol*, 5(3):233–241, Jun 1996.
- [24] Keith A. Jolley. Sequence Type Analysis and Recombinational Tests Version 2 (START2). <http://pubmlst.org/software/analysis/start2/>.
- [25] Man-Suen Chan and Nicki Ventress. STARS software.

- <http://neelix.molbiol.ox.ac.uk:8080/userweb/mchan/stars/>. STARS is an alternative interface to staden for sequence assembly for sequence typing projects.
- [26] Applied Biosystems. <http://www.appliedbiosystems.com>. SeqScape software is used for MLST.
- [27] James Gosling, Bill Joy, and Guy L. Steele. *The Java Language Specification*. Addison-Wesley Longman, 3rd edition, 2005. ISBN-10: 0321246780 ISBN-13: 978-0321246783.
- [28] Edward J Feil, Bao C Li, David M Aanensen, William P Hanage, and Brian G Spratt. eburt: inferring patterns of evolutionary descent among clusters of related bacterial genotypes from multilocus sequence typing data. *J Bacteriol*, 186(5):1518–1530, Mar 2004.
- [29] Brian G Spratt, William P Hanage, Bao Li, David M Aanensen, and Edward J Feil. Displaying the relatedness among isolates of bacterial species – the eburt approach. *FEMS Microbiol Lett*, 241(2):129–134, Dec 2004.
- [30] Katherine M E Turner, William P Hanage, Christophe Fraser, Thomas R Connor, and Brian G Spratt. Assessing the reliability of eburt using simulated populations with known ancestry. *BMC Microbiol*, 7:30, 2007.
- [31] Alexander Mellmann, Thomas Weniger, Christoph Berssenbrugge, Jorg Rothganger, Michael Sammeth, Jens Stoye, and Dag Harmsen. Based upon repeat pattern (burp): an algorithm to characterize the long-term evolution of staphylococcus aureus populations based on spa polymorphisms. *BMC Microbiol*, 7(1):98, Oct 2007.
- [32] Anca Andrei and Marcus J Zervos. The application of molecular techniques to the study of hospital infection. *Arch Pathol Lab Med*, 130(5):662–668, May 2006.
- [33] R. Guerrero. Bergey’s manuals and the classification of prokaryotes. *Int Microbiol*, 4(2):103–109, Jun 2001.
- [34] M. Hartung. *Epidemiologische Situation der Zoonosen in Deutschland im Jahr 2005 - Übersicht über die Meldungen der Bundesländer*. Bundesinstitut für Risikobewertung, 2007. ISBN 3-938163-25-9, ISSN 1614-3795.
- [35] David Livermore. The zeitgeist of resistance. *J Antimicrob Chemother*, 60 Suppl 1:i59–i61, Aug 2007.
- [36] Martin C J Maiden. Multilocus sequence typing of bacteria. *Annu Rev Microbiol*, 60:561–588, 2006.
- [37] Lance R Peterson and Stephen E Brossette. Hunting health care-associated infections from the clinical microbiology laboratory: passive, active, and virtual surveillance. *J Clin Microbiol*, 40(1):1–4, Jan 2002.

-
- [38] Larry Snyder and Wendy Champness. *Molecular Genetics of Bacteria (Second Edition)*. ASM Press, 2003.

Appendices

A PubMlst download - dbase.xml

The download of *MLST schemes* is composed of downloading sequence files (fasta formatted), a profiles (tab delimited) file and additional information, like the version. The following part of an xml file contains all necessary information for automatically downloading an *MLST scheme* of the organism “*Acinetobacter baumannii*”. The xml file is provided by PubMlst [1], at <http://pubmlst.org/data/dbases.xml>, and is used for the system feature of downloading *MLST schemes*.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<data>
<species>Acinetobacter baumannii
<mlst>
<database>
<url>http://pubmlst.org/abaumannii/</url>
<retrieved>2007-12-05</retrieved>
<profiles>
<count>20</count>
<url>http://pubmlst.org/data/profiles/abaumannii.txt</url>
</profiles>
<loci>
<locus>gltA
<url>http://pubmlst.org/data/alleles/abaumannii/gltA.tfa</url>
</locus>
<locus>gyrB
<url>http://pubmlst.org/data/alleles/abaumannii/gyrB.tfa</url>
</locus>
<locus>gdhB
<url>http://pubmlst.org/data/alleles/abaumannii/gdhB.tfa</url>
</locus>
<locus>recA
<url>http://pubmlst.org/data/alleles/abaumannii/recA.tfa</url>
</locus>
<locus>cpn60
<url>http://pubmlst.org/data/alleles/abaumannii/cpn60.tfa</url>
</locus>
<locus>gpi
<url>http://pubmlst.org/data/alleles/abaumannii/gpi.tfa</url>
</locus>
<locus>rpoD
<url>http://pubmlst.org/data/alleles/abaumannii/rpoD.tfa</url>
</locus>
</loci>
</database>
```

```
</mlst>  
</species>  
...
```


B Glossary & Acronyms

Glossary

allele

Different forms of the same gene are called alleles. 6, 9, 11, 27, 30, 40, 43, 44, 48, 49, 51, 54, 55, 75, 77, 80, 81

allele type

An allele type is a specific sequence of a gene fragment in a *MLST scheme*. Every allele type has its unique number. 6, 9, 11, 16, 17, 36, 55, 75–77

antibiotic

A chemical that inhibits the growth of an organism. It is used against infectious agents. 4, 75

assembly

A method to join fragments of sequences in order to create one sequence, the consensus sequence. 14, 17, 29, 75, 79

CLC MLST Module

The CLC MLST Module is a plug-in for CLC Workbenches (DNA and Combined Workbench). It provides functionality for *multilocus sequence typing*. 1, 25–27, 40, 45–50, 53, 55, 62, 63, 65–67, 75, 80, 81

contig

The contig (contiguous) represents overlapping *DNA* fragments originated from a single genetic source. 17, 28, 29, 36, 40, 41, 43, 51–53, 62, 63, 75, 78, 80

cross-platform

Programming languages or applications, which are cross-platform, can be used on different platforms, like Windows and Linux. 19, 20, 24, 75

Feature Driven Development

Feature Driven Development is a iterative software development process and a agile method. 21, 23, 75, 78, 80

genome

The *DNA* molecules that contain the genes for cellular growth, maintenance and metabolism, all the information to make a new organism or virus. 6, 9, 75, 76

genotype

Sequence of the *DNA* of an organism. If two organisms are genetically identical to each other they have the same genotype. 6, 75, 76

genotypic

see *genotype*. 4, 8, 9, 75

housekeeping gene

A *housekeeping gene* encodes proteins, necessary for the basic maintenance, core functions and metabolism of a cell. It is constitutively expressed, that means the expression of the gene is not regulated. 1, 75, 76

isolate

Pure culture of a bacterial species. 1–4, 6, 9, 11, 14, 16, 21, 25, 28–30, 35–38, 40, 42–44, 47, 50–55, 57–59, 62, 63, 65, 66, 75, 76, 79–81

Java

Java is an object-oriented programming language. 24, 27, 40, 75, 76

JUnit

JUnit is a framework for implementing unit tests with *Java*. 24, 75

locus

A region in the *genome* of an organism. 6, 8, 9, 12, 17, 75–77, 79

MLST scheme

A MLST scheme defines the number and lengths of the *loci* to type an organism with *MLST*. It contains *allele types*, *sequence types*, the *profiles*, as well as sequences. 6, 8–12, 21, 25, 27–30, 32–36, 38, 40, 42–45, 48–57, 59–61, 63, 73, 75, 77, 79–81

multilocus enzyme electrophoresis

A typing technique. Encoded proteins of an organism are used to detect enzyme polymorphisms on basis of differing electrophoretic mobilities of those proteins on a gel. 8, 75, 78

multilocus sequence analysis

Analyses based on the data used with *MLST* are called *Multilocus sequence analysis*. 6, 7, 14, 17, 21, 25, 65, 75, 76, 78, 79

multilocus sequence typing

A typing method to distinguish different isolates of a microorganism. Multilocus means that up to 11 *loci* of an *isolate* are used to determine a *sequence type*. Fragments of those *loci* are compared and arbitrarily numbers are given to alleles, the *allele types*. The difference of more than one nucleotide gives a new *allele type*. The combination of *allele types* of the *loci* gives the *sequence type*. 1, 2, 6, 7, 14–19, 21, 25, 29, 40, 53, 62, 65–67, 75, 77–79

phenotype

All observable properties of an organism. 75, 77

phenotypic

see *phenotype*. 4, 8, 9, 75

Polymerase chain reaction

A method of molecular biology to exponentially copy a fragment of a sequence in vitro (outside of an organism). 6, 14, 75, 78

profile

The profile is a combination of numbers for genes in a *MLST scheme*, which represents the *sequence type*. Every gene is associated with a number and this number links to a sequence in the *MLST scheme*, the *allele* sequence. The profile was developed by reason of human readability. 9, 11, 12, 27, 28, 43, 44, 54, 59, 75, 76, 79–81

pulsed-field gel electrophoresis

A typing technique. Rare-cutting restriction enzymes are used to get several fragments of a chromosomal *DNA* sequence. Those fragments are separated on an agarose gel with periodically changing the direction of the electrical field. 8, 75, 78

sequence type

A sequence type is a specific arrangement of *allele types*, a specific sequence in all *loci* of a *MLST scheme*. Every sequence type has its unique number. 6, 11, 12, 14–17, 20, 27, 28, 35, 36, 44, 51, 52, 54, 55, 59, 63, 75–77, 79, 81

singlelocus sequence typing

Same principle as *multilocus sequence typing*, however, just one locus is used in this typing method. 8, 75, 77

Spa-typing

Spa-typing is a rapid *singlelocus sequence typing* method of the organism *Staphylococcus aureus* (also *MRSA*). There is a gene called “spa” in *MRSA* which is used for typing. 8, 75

strain

A variant of an organism. 75

Test Driven Development

The test-driven development is a software development technique. The implementation of new code or a new feature starts with implementing tests. After implementing the tests the code is implemented. Only if the tests succeed, the new code or feature is accepted. The code is refactored and the cycle starts again with new code. All tests, also tests of other features, have to succeed on every end of a cycle. 22–24, 75, 79, 80

trace

The term “trace” is applied for sequence data quality in this thesis. Automated *DNA* sequencing instruments use fluorescently labelled samples resulting in

fluorescent peak trace chromatograms. Two sequences, with their trace data attached, are shown in the *contig* of the Figure 6.. 11, 63, 75, 81

typing

Typing means to

- characterize an organism
- distinguish between different isolates of an organism

. 3, 18, 21, 28–31, 36, 37, 39, 42, 52, 63, 75, 80

Acronyms

bp

base pair. 9, 75

DNA

Deoxyribonucleic acid. 1, 4, 6, 8, 14, 21, 63, 75, 77

FDD

Feature Driven Development. 21–23, 75, 80

kbp

kilo base pairs. 8, 75

MLEE

Multilocus enzyme electrophoresis. 8, 9, 75, 82

MLSA

Multilocus sequence analysis. 75

MLST

Multilocus sequence typing. 1, 2, 6–9, 14, 18, 19, 25, 27, 29, 39, 62, 63, 65–67, 75, 76, 79, 82

MRSA

Methicillin-resistant *Staphylococcus aureus*. 8, 75, 77

PCR

Polymerase chain reaction. 14, 75

PFGE

Pulsed-field gel electrophoresis. 8, 9, 75, 82

RNA

Ribonucleic acid. 4, 75

SOAP

Simple Object Access Protocol. 66, 75

ST*Sequence type*. 11, 12, 75, 79**TDD***Test Driven Development*. 22, 23, 75, 80**C List of Figures**

1	CLC bio's solutions	
	Main segments of CLC bio's [6] offers are software, consulting and high performance computing. The software segment is made up of solutions like the Combined (yellow), Protein (green), DNA (blue) and RNA (purple) Workbench, as well as the Software Developer Kit (golden) and the Plug-ins (black), as at November 2007.	5
2	Overview of work regarding <i>MLST</i> . The result of the preparation step and <i>multilocus sequence typing</i> will be added to a database and is stored for more analyses, the <i>Multilocus sequence analyses</i>	7
3	Example of some data of a <i>MLST scheme</i> , the <i>Staphylococcus aureus</i> from Mlst.net [19].	10
4	The connection between the <i>profile</i> and sequences is shown in this figure. Every number in the <i>profile</i> represents a sequence in the same <i>MLST scheme</i> . The combination of the numbers makes up a <i>sequence type</i> . All isolates of same <i>sequence type</i> must have the same sequence in every <i>locus</i> . The <i>sequence type</i> box contains a circular illustration of a bacterial chromosome, which comprises the sequences of an organism of <i>ST</i> 8.	12
5	Online available PubMlst's <i>sequence type</i> query for organism <i>Streptococcus uberis</i> : Figure 5a is showing the input form where you enter the sequences. Figure 5b is an example result. It is the query result if the input sequences are from a <i>sequence type</i> 1 isolate.	15
5	(Continued) Online available PubMlst's <i>sequence type</i> query for organism <i>Streptococcus uberis</i> : Figure 5c is an example result. It is an <i>isolate</i> , which did not match with a <i>sequence type</i> in the database and probably is a new <i>sequence type</i>	16
6	A contig: The <i>assembly</i> of two sequences (forward read and backward read) is shown. The result is the consensus sequence, which is used for further analyses.	17
7	Overview of work regarding <i>MLST</i> . Contrary to Figure 2 the tasks are red which can easily be performed automatically.	18
8	The Feature Driven Development software process. Source: Nebulon Pty. Ltd.	22

9	<i>FDD-TDD</i> approach of this project. The “Build by feature” process of <i>Feature Driven Development</i> is extended with <i>Test Driven Development</i> . Every feature implementation starts with implementing tests. After finishing the tests, the functionality can be implemented. The tests have to succeed, before passing over to refactoring. The “Build by feature” step is iterative, that means the feature can be partitioned before the implementation starts.	23
10	A data flow diagram, representing the overall features, which will be integrated in the <i>CLC MLST Module</i> . Arrows represent data flow, circles imply features. The “User” is an interface and the “MLST data file” the data storage.	26
11	An overall class diagram of the most important classes in the <i>CLC MLST Module</i>	27
12	Use case of system feature “Typing workflow”. Main actors are the user and the system. The subsystem ‘Typing’ represents the main functionality for <i>typing</i>	31
13	Use case of system feature “Create <i>MLST scheme</i> ”.	32
14	Use case of system feature “Download <i>MLST scheme</i> ”.	33
15	Use case of system feature “Merge <i>MLST schemes</i> ”.	34
16	Use case of system feature “Extend MLST scheme”	36
17	Use case of system feature “Extend Isolate”. The subsystem “ <i>Typing</i> ” of the system feature “Typing workflow” in Section 5.6.2, Page 29, is reused.	37
18	Use case of system feature “Submission of data”.	39
19	Data classes introduced with the MLST module. Some classes like the <i>Contig</i> and the <i>SequenceList</i> have already been introduced with the CLC bio software and are reused for the MLST module project. .	41
20	Activity diagram of the automatic assignment	42
21	This activity diagram visualises the algorithm for the determination of the <i>allele profile</i> . The first iteration (green) gets the sequences for each gene of the <i>isolate</i> . This sequence can come from a <i>contig</i> object or from a sequence object. This sequence is then compared to <i>allele</i> sequences of the <i>MLST scheme</i> in the second iteration (blue). If the sequence matches with an <i>allele</i> sequence of the <i>MLST scheme</i> the <i>profile</i> number of the <i>allele</i> sequence is fetched and assigned to the <i>isolate</i> ’s sequence. If there is no match the <i>isolate</i> ’s sequence is a new <i>allele</i> sequence. The combination of the numbers determined in this algorithm represents the <i>profile</i> of the <i>isolate</i>	43

22	The activity diagram shows the determination of a list of possible <i>sequence types</i>, the typing algorithm . As soon as the <i>profile</i> of the <i>isolate</i> contains an entry with a negative number (this indicates a new <i>allele</i> type), the <i>sequence type</i> list will be extended with an entry of a 'New <i>sequence type</i> '. The <i>isolate profile</i> is compared to the entries of the <i>profile</i> list of the referenced <i>MLST scheme</i> . If the <i>profile</i> entries match, the <i>sequence type</i> of the <i>MLST scheme</i> 's <i>profile</i> entry is determined and added to the <i>sequence type</i> list. Not assigned gene entries of the <i>isolate</i> will match with every entry, so the user gets a list of possible <i>sequence types</i>	44
23	Dialog windows of the action "Assembly and Create Isolate" are shown by screenshots of the <i>CLC MLST Module</i> 1.0. The selection of the input objects (23a) is the first step. Afterwards, the <i>MLST scheme</i> , input parameters for the alignment and the trimming can be set (23b).	45
23	(Continued) Dialog windows of the action "Assembly and Create Isolate" are shown by screenshots of the <i>CLC MLST Module</i> 1.0. The Figure 23c reveals the automatic assignment of a gene. Users have the option of opening or saving the result immediately (Figure 23d).	46
23	(Continued) Dialog windows of the action "Assembly and Create Isolate" are shown by screenshots of the <i>CLC MLST Module</i> 1.0. The <i>Isolate</i> object comprises results and parameters of the action.	47
24	Dialog windows of the action "Create <i>MLST scheme</i> " are shown by screenshots of the <i>CLC MLST Module</i> 1.0.	49
25	The download dialog window of the <i>CLC MLST Module</i> is shown in this figure. Some of the available organisms of PubMlst [1] are represented within the screenshot.	50
26	The automatic update dialog window is activated with opening an <i>isolate</i> and differing versions of the <i>isolate</i> and its referenced <i>MLST scheme</i>	54
27	History entries are shown in a history view in every object in the CLC Workbench. This is a <i>MLST scheme</i> representing some changes. . . .	56
28	The main view of the <i>Isolate</i> object. All assigned objects are shown for the genes as well as the referenced <i>MLST scheme</i>	57
29	This figure represents the view of all collected informations of the typing for this <i>isolate</i> , the <i>Isolate</i> report view.	58
30	This figure represents the view of a <i>MLST scheme</i> object with its allele sequences in tables.	60
31	This view of a <i>MLST scheme</i> shows the profiles contained in the object.	61
32	Mlst workflow. This figure shows the difference of time spent for typing an <i>isolate</i> with the <i>CLC MLST Module</i> and an online form, e.g. PubMlst [1].	62
33	Evaluation of <i>trace</i> files, provided by PubMlst [1], for organism <i>Streptococcus uberis</i>	63

D List of Tables

- | | | |
|---|--|----|
| 1 | Most commonly used typing techniques in molecular biology. Attributes, explained in Section 2.3, are compared of <i>MLST</i> , <i>MLEE</i> and <i>PFGE</i> | 9 |
| 2 | Public available databases with organisms as at end of October 2007. | 13 |