Aalto University
School of Science
Degree Programme in Computer Science and Engineering

Stella Student

# Software Processes for Dummies:
## Re-inventing the Wheel

Master's Thesis
Espoo, June 18, 2011

**DRAFT! — August 9, 2016 — DRAFT!**

Supervisors: Professor Antti Ylä-Jääski, Aalto University
Professor Pekka Perustieteilijä, University of Helsinki
Advisor: Olli Ohjaaja M.Sc. (Tech.)

Aalto University
School of Science
Degree Programme in Computer Science and Engineering

ABSTRACT OF
MASTER'S THESIS

**Author:** Stella Student

**Title:**
Software Processes for Dummies: Re-inventing the Wheel

| | | | |
|---|---|---|---|
| **Date:** | June 18, 2011 | **Pages:** | 38 |
| **Major:** | Data Communication Software | **Code:** | T-110 |
| **Supervisors:** | Professor Antti Ylä-Jääski<br>Professor Pekka Perustieteilijä | | |
| **Advisor:** | Olli Ohjaaja M.Sc. (Tech.) | | |

A dissertation or thesis is a document submitted in support of candidature for a degree or professional qualification presenting the author's research and findings. In some countries/universities, the word thesis or a cognate is used as part of a bachelor's or master's course, while dissertation is normally applied to a doctorate, whilst, in others, the reverse is true.

!FIXME **Abstract text goes here (and this is an example how to use fixme).** FIXME! Fixme is a command that helps you identify parts of your thesis that still require some work. When compiled in the custom `mydraft` mode, text parts tagged with fixmes are shown in bold and with fixme tags around them. When compiled in normal mode, the fixme-tagged text is shown normally (without special formatting). The draft mode also causes the "Draft" text to appear on the front page, alongside with the document compilation date. The custom `mydraft` mode is selected by the `mydraft` option given for the package `aalto-thesis`, near the top of the `thesis-example.tex` file.

The thesis example file (`thesis-example.tex`), all the chapter content files (`1introduction.tex` and so on), and the Aalto style file (`aalto-thesis.sty`) are commented with explanations on how the Aalto thesis works. The files also contain some examples on how to customize various details of the thesis layout, and of course the example text works as an example in itself. Please read the comments and the example text; that should get you well on your way!

**Keywords:** ocean, sea, marine, ocean mammal, marine mammal, whales, cetaceans, dolphins, porpoises

**Language:** English

Aalto-yliopisto
Perustieteiden korkeakoulu
Tietotekniikan koulutusohjelma

DIPLOMITYÖN
TIIVISTELMÄ

| | | | |
|---|---|---|---|
| **Tekijä:** | Stella Student | | |
| **Työn nimi:** | Ohjelmistoprosessit mänteille: Uusi organisaatio, uudet pyörät | | |
| **Päiväys:** | 18. kesäkuuta 2011 | **Sivumäärä:** | 38 |
| **Pääaine:** | Tietoliikenneohjelmistot | **Koodi:** | T-110 |
| **Valvojat:** | Professori Antti Ylä-Jääski<br>Professori Pekka Perustieteilijä | | |
| **Ohjaaja:** | Diplomi-insinööri Olli Ohjaaja | | |

Kivi on materiaali, joka muodostuu mineraaleista ja luokitellaan mineraalisisältönsä mukaan. Kivet luokitellaan yleensä ne muodostaneiden prosessien mukaan magmakiviin, sedimenttikiviin ja metamorfisiin kiviin. Magmakivet ovat muodostuneet kiteytyneestä magmasta, sedimenttikivet vanhempien kivilajien rapautuessa ja muodostaessa iskostuneita yhdisteitä, metamorfiset kivet taas kun magma- ja sedimenttikivet joutuvat syvällä maan kuoressa lämpötilan ja kovan paineen alaiseksi.

Kivi on epäorgaaninen eli elottoman luonnon aine, mikä tarkoittaa ettei se sisällä hiiltä tai muita elollisen orgaanisen luonnon aineita. Niinpä kivestä tehdyt esineet säilyvät maaperässä tuhansien vuosien ajan mätänemättä. Kun orgaaninen materiaali jättää jälkensä kiveen, tulos tunnetaan nimellä fossiili.

Suomen peruskallio on suurimmaksi osaksi graniittia, gneissiä ja Kaakkois-Suomessa rapakiveä.

Kiveä käytetään teollisuudessa moniin eri tarkoituksiin, kuten keittiötasoihin. Kivi on materiaalina kalliimpaa mutta kestävämpää kuin esimerkiksi puu.

| | |
|---|---|
| **Asiasanat:** | AEL, aineistot, aitta, akustiikka, Alankomaat, aluerakentaminen, Anttolanhovi, Arcada, ArchiCad, arkki |
| **Kieli:** | Englanti |

Aalto-universitetet
Högskolan för teknikvetenskaper
Examensprogram för datateknik

SAMMANDRAG AV
DIPLOMARBETET

| | | | |
|---|---|---|---|
| **Utfört av:** | Stella Student | | |
| **Arbetets namn:** | Den stora stygga vargen: Lilla Vargens universum | | |
| **Datum:** | Den 18 Juni 2011 | **Sidantal:** | 38 |
| **Huvudämne:** | Datakommunikationsprogram | **Kod:** | T-110 |
| **Övervakare:** | Professor Antti Ylä-Jääski<br>Professor Pekka Perustieteilijä | | |
| **Handledare:** | Diplomingenjör Olli Ohjaaja | | |

Lilla Vargens universum är det tredje fiktiva universumet inom huvudfåran av de tecknade disneyserierna - de övriga två är Kalle Ankas och Musse Piggs universum. Figurerna runt Lilla Vargen kommer huvudsakligen från tre källor — dels persongalleriet i kortfilmen Tre små grisar från 1933 och dess uppföljare, dels långfilmen Sången om Södern från 1946, och dels från episoden "Bongo" i långfilmen Pank och fågelfri från 1947. Framför allt de två första har sedermera även kommit att leva vidare, utvidgas och införlivas i varandra genom tecknade serier, främst sådana producerade av Western Publishing för amerikanska Disneytidningar under åren 1945–1984.

Världen runt Lilla Vargen är, i jämförelse med den runt Kalle Anka eller Musse Pigg, inte helt enhetlig, vilket bland annat märks i Bror Björns skiftande personlighet. Den har även varit betydligt mer öppen för influenser från andra Disneyvärldar, inte minst de tecknade långfilmerna. Ytterligare en skillnad är att varelserna i vargserierna förefaller stå närmare sina förebilder inom den verkliga djurvärlden. Att vargen Zeke vill äta upp grisen Bror Duktig är till exempel ett ständigt återkommande tema, men om katten Svarte Petter skulle få för sig att äta upp musen Musse Pigg skulle detta antagligen höja ett och annat ögonbryn bland läsarna.

| | |
|---|---|
| **Nyckelord:** | omsättning, kassaflöde, värdepappersmarknadslagen, yrkesutövare, intresseföretag, verifieringskedja |
| **Språk:** | Engelska |

# Acknowledgements

I wish to thank all students who use LaTeX for formatting their theses, because theses formatted with LaTeX are just so nice.

Thank you, and keep up the good work!

Espoo, June 18, 2011

Stella Student

# Abbreviations and Acronyms

| | |
|---|---|
| 2k/4k/8k mode | COFDM operation modes |
| 3GPP | 3rd Generation Partnership Project |
| ESP | Encapsulating Security Payload; An IPsec security protocol |
| FLUTE | The File Delivery over Unidirectional Transport protocol |
| e.g. | for example (do not list here this kind of common acronymbs or abbreviations, but only those that are essential for understanding the content of your thesis. |
| note | Note also, that this list is not compulsory, and should be omitted if you have only few abbreviations |

# Contents

# Chapter 1

# Introduction

This is my master's thesis, and I am very proud of it. Of course, when I write my *real* master's thesis, I will not use the singular pronoun *I*, but rather try to avoid referring to myself and speak of the research *we* have conducted—I rarely work alone, after all. Yet, both *I* and *we* are correct, and it depends on the instructor and the supervisor (of course from you, too), which one they would prefer. Anyway, the tense should be active, and passive sentenses should be avoided (especially, writing sentences where the subject is presented with by preposition), so often you cannot avoid choosing between the pronouns. Life is strange, but there you have it.

By the way, the preferred order of writing your master's thesis is about the same as the outline of the thesis: you first discover your problem and write about that, then you find out what methods you should use and write about that. Then you do your implementation, and document that, and so on. However, the abstract and introduction are often easiest to write last. This is because these really cover the entire thesis, and there is no way you could know what to put in your abstract before you have actually done your implementation and evaluation. Rarely anyone write the thesis from the beginning to the end just one time, but the writing is more like process, where every piece of text is written at least twice. Be also prepared to delete your own text. In the first phase, you can hide it into comments that are started with % but during the writing, the many comments should be visible for your helpers, the instructor and supervisor.

The introduction in itself is rarely very long; two to five pages often suffice.

## 1.1 Problem statement

Undergraduate students studying technical subjects do not consider typography very interesting these days, and therefore the typographical quality of many theses is unacceptably low. We plan to rectify this situation somewhat by providing a decent-quality example thesis outline for students. We expect that the typographical quality of the master's theses will dramatically increase as the new thesis outline is taken into use.

## 1.2 Helpful hints

Read the information from the university master's thesis pages [4] before starting the thesis. You should also go through the thesis grading instructions [1] together with your instructor and/or supervisor in the beginning of your work.

## 1.3 Structure of the Thesis

You should use transition in your text, meaning that you should help the reader follow the thesis outline. Here, you tell what will be in each chapter of your thesis.

# Chapter 2

# Background

The problem must have some background, otherwise it is not interesting. You can explain the background here. Probably you should change the title to something that describes more the content of this chapter. Background consists of information that help other masters of the same degree program to understand the rest of the thesis.

Transitions mentioned in Section 1.3 are used also in the chapters and sections. For example, next in this chapter we tell how to use English language, how to find and refer to sources, and enlight different ways to include graphics in the thesis.

## 2.1 Language and Structure

Moreover, the transitions are also used in the paragraph and the sentence level meaning that all the text is linked together. For example, the word "moreover" here is one way, but of course you should use variation in the text. Examples of transitional devices (words) and their use can be found from writing guides, e.g. from the Online Writing Lab (OWL)[1] of Purdue University or Strunk's Elements of Style[2]. Remember that footnotes are additional information, and they are seldom used. If you refer to a source, you do no not use footnote. The right command for the references is *cite*.

The language used in the thesis should be technical (or scientifical). For example, the abreviations aren't used but all them are written open (i.e. "are not"). Since the content itself is often hard to understand (and explain), the sentences should not be very long, use complex language with several examples embedded in the same sentence, and, also, seldom used words and

[1] http://owl.english.purdue.edu/owl/resource/574/02/

[2] http://www.bartleby.com/141/

weird euphemism or paraphrases can make the sentence hard to follow and to read it with only one time, and making everything even harder to understand all this without any punctuation marks makes the instructor cry and finally after trying to correct the language, you will get boomerang, and everyone's time has just been wasted.

Please use proofreaders before sending even your unfinished version to the instructor and/or supervisor. You will get better comments when they do not need first proofread your text. Moreover, they can consentrate to the content better if the language and spelling mistakes are not distracting the reading. Several editors have their own proofreading tools, e.g. ispell in emacs. You can also use Microsoft Word to proofread your thesis: it can correct also some grammatical errors and not just misspelled words.

Note also that if you have a section or a subsection, you have to have at least two of them, or otherwise the section or subsection title is unnecessary. Same with the paragraphs an: you should not have sections with only one paragraphe, and single sentence paragraph. Furthermore, always write some text after the title before the next level title.

## 2.2 Finding and referring to sources

Never ever copy anything into your theses from somebody else's text (nor your own previously published text). Never. Not even for starting point to be rewritten later. The risk is that you forgot the copied text to your thesis and end up to be accused of plagiarism. Plagiarism is a serious crime in studies and science and can ruin your career even its beginning. To repeate: never cut and paste text into your thesis!

### 2.2.1 Finding sources

All work is based on someone else's work. You should find the relevant sources of your field and choose the best of them. Also, you should refer to the original source where a fact has been mentioned first time. Remember source evaluation (criticism) with all sources you find.

Good starting points for finding references in computer science are:

- Nelli Portal (Aalto Library): `http://www.nelliportaali.fi`
- ACM Digital library: `http://portal.acm.org/`
- IEEExplore: `http://ieeexplore.ieee.org`
- ScienceDirect: `http://www.sciencedirect.com/`

- ... although Google Scholar (`http://scholar.google.com/`) will find links to most of the articles from the abovementioned sources, if you search from within the university network

Some of the publishers do not offer all the text of the articles freely, but the library has agreed on the rights to use the whole text. Thus, you should sometimes use computers in the domain of the university in order to get the full text. Sometimes the Nelli Portal can also help getting the whole article instead of just the abstract. The library has also brief instrucions how to find information [2].

Instead of normal Google, use Google Scholar (`http://scholar.google.fi/`). It finds academic publications whereas normal Google find too much commercial advertisements or otherwise biased information. Wikipedia articles should be referred to in the master thesis only very, very seldomly. You can use Wikipedia for understanding some basics and finding more sources, but often you cannot be sure if the article is correct and unbiased.

One important part of the sources that you have found is the reference list. This way you can find the original sources that all the other research of the field refer. Often you can also find more information with the name of the researchers that are often referred in the articles.

### 2.2.2 Referring to sources

The main point in referring to sources is to separate your own thinking and text from that of others. Facts of the research area can be given without reference, but otherwise you should refer to sources. This means two things: marking the source in the text where it has been used, and listing the sources usually in the end of the thesis in a way that help the reader to find the original source.

There are several bibliography styles, meaning how to form the bibliography in the end of the thesis. Aalto's library has good instructions for many styles [3]. You should ask from your supervisor or instructors which style you should use. This thesis template uses the number style that is often used in software engineering. The other style also used in the CS field, e.g. usability, is the Harvard style where instead of numbers, the reference is marked into the text with author's name and publishing year. Other areas use also many other styles for making the lists and marking the references.

In addition to the list in the end of the thesis, you have to mark the source in the text where the source is used. There are three places for the reference: in a sentence before the period, in the end of a sentence after the period, or in the end of a paragraph. All of them have different meaning.

The main point is that first you paraphrase the source using your own words and then mark the source. Next, we give short examples that are marked with *emphasised text*.

*Haapasalo [10] researched database algorithms that allows use of previous versions of the content stored in the database.* This kind of marking means that this paragraph (or until the next reference is given) is based on the source mentioned in the beginning. Giving the source you should use only the family name of the first author of the article, and not give any hints about what is the type of the article that is referred.

*B+-trees offers one way to index data that is stored in to a database. Multiversion B+-trees (MVBT) offer also a way to restore the data from previous versions of the database. Concurrent MVBT allows many simultaneous updates to the database that is was not possible with MVBT. [10]* When the marking is after the period, the reference is retrospective: all the paragraph (or after previous reference marking) is based on the source given in its end. If the content is very broad, you can start with saying *According to Haapasalo*, then continue referring the source with several separate sentences, and in the end put the marking of your source *that shows that CMVBT are the best. [10].*

If your paragraph has several sources, the above mentioned styles are not proper. The reader of your thesis cannot know which of your sources give which of the statements. In this case, it is better to use more finegraded refering where the reference markings that are embedded in the sentences. For example, *the multiversion B+-tree (MVBT) index of Becker et al. [6] allows database users to query old versions of the database, but the index is not transactional. It's successor, the transactional MBVT (TMVBT), allows a single transaction running in its own thread or process to update the database concurrently with other transactions that only read the database [11]. Further development, titled the concurrent MBVT (CMVBT), allows several transactions to perform updates to the database at the same time [10].* Here, the references are marked before the period in the sentences where they are used.

Finally, direct quotes are allowed. However, often you should avoid them since they do not usually fit in to your text very well. Using direct quotes has two tricks: quotation marks and the source. *"Even though deletions in a multiversion index must not physically delete the history of the data items, queries and range scans can become more efficient, if the leaf pages of the index structure are merged to retain optimality." [10]* Quotes are hard to make neatly since you should use only as much as needed without changing the text. Moreover, you often do not really understand what the author has mentioned with his wordings if you cannot write the same with your own words. Remember also that never cut and paste anything without marking

the quotation marks right away, and in general, never cut and paste anything at all!

Sometimes getting the original source can be almost impossible. In an extremely desperate situation, you can refer with structure *mr X [...] according to ms Y [...] defined that*, if you find a source that refers to the original source. Note also that the reference marking is never used as sentence element (example of how **not** to do it: *[10] describes an optimal algorithm for indexing multiversiond databases.*).

# Chapter 3

# Clustering Methods

As described in Chapter XX, all patients data is retrieved from the database directly without any manual labeling. Due to lack of labels, unsupervised learning algorithms should be adopted. Clustering is a collection of unsupervised methods, which identifies groups of data points according to a defined similarity metric, such that objects in the same group posses higher similarities compared to objects in other groups. The clustering process does not rely on labels but the choice of similarity metrics. Variations in similarity metrics lead to different clustering methods.

Applying clustering methods in anomaly detection tasks has been studied numerously [12] . This chapter introduces two typical methods, K-Means [13] and DBSCAN [8]. Problem formulation, solutions, and potential issues are formally described using elaborated notations in following sections. However, analysis on the performance and constraints of these two methods are postponed to Chapter XXX, which reveals their practicality and infeasibility in the previously described anomaly detection problem. (Time, Memory consumption. For K-Means, definition of center.)

## 3.1 K-Means

K-Means is one of the simplest unsupervised algorithm which solves clustering problem. Despite its simplicity, K-Means has gained success in various situations, including anomaly detection [12] [7], image segmentation and compression [**?** ], and preprocessing for more complicated algorithms. The method can be formally defined as follow: Given a data set $\{\mathbf{x}_1, ..., \mathbf{x}_N\}$ consisting of $N$ observations in $D$-dimensional space, the object is to partition the data into $K$ groups, by defining a set of $K$ centers $\{\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_k\}$ in the same space, and assigning each observation to exactly one center point. Each

center point represents a prototype associated with the $k^{th}$ cluster.

The assignments can be represented using 1-of-$K$ scheme. Then, for each data point $x_n$, a corresponding $K$-dimensional variable consisting of $K$ binary elements $r_{nk} \in \{0, 1\}$ is introduced. Among these $r_{nk}$, exactly one of them equals 1, which means $\mathbf{x}_n$ belongs to the $k^{th}$ cluster. Using this notation, evaluation of the clustering quality can be defined using the object function as follow:

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} D(\mathbf{x}_N - \boldsymbol{\mu}_k) \tag{3.1}$$

where $D$ is the dissimilarity metric. Common choice of the metric is $l_1$-norm or $l_2$-norm. Mahalanobis distance is also adopted while considering the covariances between the $K$-dimensions [**?** ]. In the following context, $l_2$-norm is adoped for discussion. Intuitively, this function can be considered as the distance summation of each point to its corresponding cluster prototype $\boldsymbol{\mu}_k$. The K-Means aims at finding a set of $\boldsymbol{\mu}_k$ which minimizes the object function. Finding the optimal solution for the above object function proves to be NP-Hard [5]. However, employing heuristic algorithms enables finding converged local optimal solutions. Section 3.1.1 describes one iterative algorithm, EM. Section 3.1.2 explores common issues related to K-Means and remedies.

### 3.1.1 EM Algorithm in K-Means

EM algorithm is an iterative algorithm to find local maximum. Each iteration consists of two phases, Expectation and Maximization, which corresponds to minimize the objective function $J$ with respect to $r_{nk}$ and $\boldsymbol{\mu}_k$ respectively. In the E(expectation) step, the algorithm minimize $J$ with respect to $r_{nk}$, while keeping the $\boldsymbol{\mu}_k$ fixed. Then in the following M(maximization) step, the algorithm minimizes $J$ with respect to $\boldsymbol{\mu}_k$, while keeping $r_{nk}$ fixed.

Considering the optimization in E step, a critical observation of (3.1) is that $\mathbf{r}_n$'s are independent of each other. Thus, optimization on $\mathbf{r}_n$'s can be done seperately for each $n$. The solution is simply setting the $r_{nk}$ corresponding to the minimum $\|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$ to 1. Intuitively, the algorithm assigns $\mathbf{x}_n$ to the closest cluster center. Formally, the solution can be written as

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \| \mathbf{x}_n - \mu_j \|^2 \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

In the following M step, the above determined $r_{nk}$ is clamped. Then, $\boldsymbol{\mu}_k$ appears only in a quadratic term in $J$. Setting derivatives of $J$ with respect to $\boldsymbol{\mu}_k$ to zero, solution formula for $\boldsymbol{\mu}_k$ can be expressed in following closed

form

$$\boldsymbol{\mu}_k = \frac{\sum_n r_{nk}\mathbf{x}_n}{\sum_n r_{nk}} \tag{3.3}$$

This step can be considered as recomputing the cluster prototype by setting $\boldsymbol{\mu}_k$ to the mean of all points assigned to that cluster.

EM keeps executing these two steps alternatively, until a convergence happens or until number of iterations exceeded a predifined value. Since in each phase, one variable is fixed and updating another variable minimizes the cost function $J$, convergence is guaranteed. Formal proof on convergence has been studied by MacQueen [**?** ].

The algorithm is illustrated using dataset generated independently from three gaussian distributions in Figure 3.1. In this demonstration, the algorithm takes $K = 3$, which is the correct number of components. Before running the first iteration, initialized $\boldsymbol{\mu}_k$ is required. This initialization is done by choosing three objects from the data set randomly.

### 3.1.2 Issues in K-Means

Despite the simplicity of K-Means, several underlying issues exists. The first potential is that, how to choose the value for $K$. In the above illustration, $K$ was set to 3 which is the correct number of components. However, if $K$ wasn't set to the correct value, unsatisfied clustering may be generated. Example of this issue is shown in Figure 3.2(a)-(c). To solve this problem, one practical way is drawing graph of the cost function versus value of $K$, as shown in Figure 3.2(d). Intuitively, when $K$ is under smaller than the true number of clusters, increasing $K$ will lead to a huge drop of cost function value. However, when $K$ has reached or exceeded the correct value, incresing $K$ leads only small cost function value drop. Thus, number of clusters corresponding to the 'elbow' point can be considered as the real number of clusters.

Another issues of K-Means relates to the initialization of $\boldsymbol{\mu}_k$. Since EM finds only local optimal solutions, a poor initialization could lead to worse clustering result. To avoid this problem, it is practical to run K-Means for several times and choose the best result according to the value of the cost function.

One more critical issue of K-Means lies in the choice of similarity metric. As mentioned at the beginning of this chapter, variations in similarity metric leads to different clustering methods. Choosing $l$-2 norm is convenient in terms of computation, but this choice limits the type of data variables to certain types. Using $l$-2 norm on categorical data is inappropriate since no ordering of categorical values exists. Also, this makes computing the mean

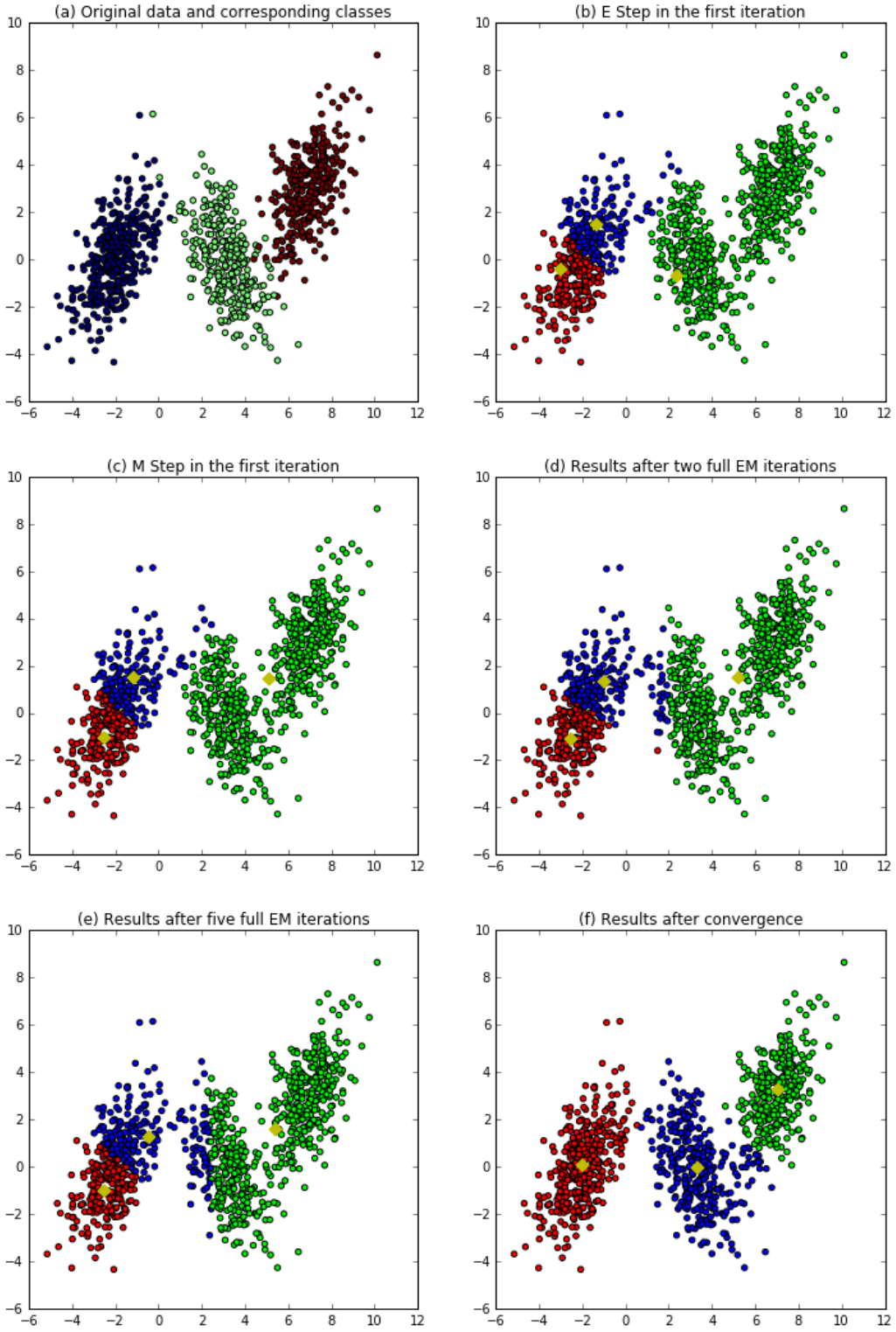



Figure 3.1: Illustraion of EM algorithm on K-Means using data generated independently from three gaussian distribution. (a) Original data and corresponding classes. Classes are denoted in different colors. (b) Assignments of each data after the first E step. The yellow diamonds represent the intialized $\boldsymbol{\mu}_k$. (c) Updated $\boldsymbol{\mu}_k$ after the M step in the first iteration. (d)-(f) Clustering results after several successive full EM iterations until convergence is met.

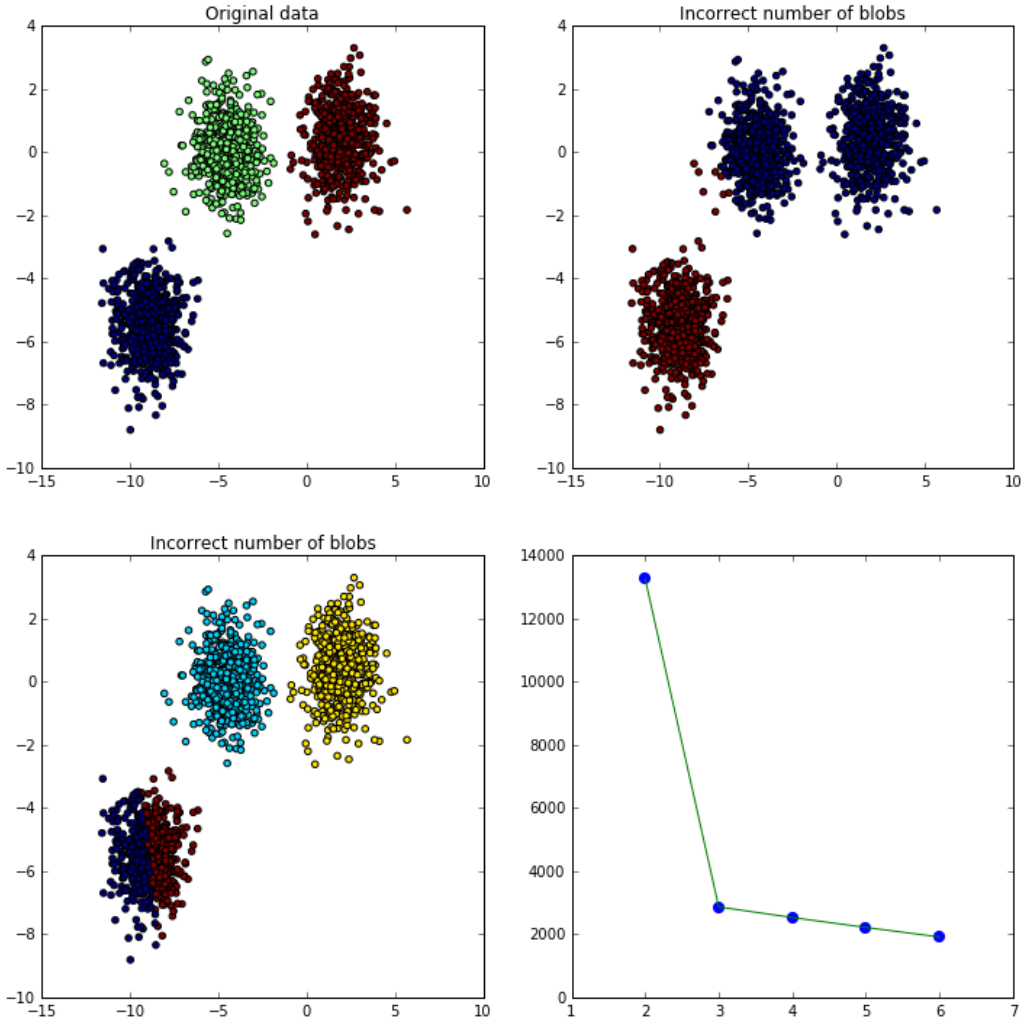



Figure 3.2: Issues of K-Means incurred by choosing inappropriate value for $K$

value a hard problem. To use K-Means on other data types, the similarity metric should be elaborately designed.

Besides, K-Means tends to form clusters into a convex space. As shown in Figure 3.1(b)-(e), the boundary between two different clusters forms a line lying at the midway and is perpendicular to the line connecting two cluster prototypes. However, a cluster is not necessary to be convext. This also limits the application of K-Means.

Finally, K-Means is also sensitive to noises. As shown in formula 3.3, updating $\boldsymbol{\mu}_k$ involves computing the mean of all data points assigned to that cluster. When noise objects with large deviation from other points in this cluster exist, update of $\boldsymbol{\mu}_k$ will be strongly affected.

## 3.2 Density Based Clustering Methods

This section explores another type of clustering method which forms cluster from the view of density aspect. **D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise (DBSCAN) [8] considers to be one of the most successful method from this category [**?** ]. Compared to K-Means, DBSCAN posseses following advantages: 1). DBSCAN can detect clusters of arbitrary shape. 2). DBSCAN can determine the number of clusters automatically. 3). DBSCAN is robust to noises.

The following of this chapter is organized as follows. Section 3.2.1 compares and reveals the relation of DBSCAN to a basic graph traversal algorithm **D**epth **F**irst **S**earch(DFS). This section will introduce the critical concept used by DBSCAN which differentiate it with DFS. Section 3.2.2 describe this concept in more details. Section 3.2.3 discusses practial issues to accelerate DBSCAN.

### 3.2.1 DBSCAN and DFS

Before introducing DBSCAN, it would be helpful to review a basic graph traversal algorithm, DFS, which is an algorithm which traverses or visits a graph. In the discussion of this context, the whole process is divided into two procedures DFS and DFS-Visit. A typical recursive implementation is described as follow.

DFS($G$)

```
1  for each vertex u ∈ G
2      u.visited = FALSE
3  for each vertex u ∈ G
4      if u.visited == FALSE
5          DFS-VISIT(G, u)
```

DFS-VISIT($G, u$)

```
1  u.visited = TRUE
2  for each vertex v ∈ G.Adj[u]
3      if v.visited == FALSE
4          DFS-VISIT(G, v)
```

The first two lines in DFS initializes the algorithm by setting all nodes to be unvisited. Then, the algorithm traverse vertices in the graph. Once an unvisited node is found, DFS-VISIT is called. In the procedure DFS-VISIT, the given node $u$ is labled as visited. If an unvisited child $v$ of $u$ is found, this procedure goes one layer deeper by calling another DFS-VISIT on $v$. After finishing DFS-VISIT($G, u$), the algorithm backtrack to visit other unvisited children of $u$, which are siblings of $v$.

Assume DFS executes on an undirected graph. Each call of DFS-VISIT on an unvisited node $u$ explores $u$ and all nodes reachable to it. These nodes form a component which disconnects with other components by other runs of DFS-VISIT. Thus, each component can be considered as a cluster. The criterion to form a cluster is that each pair of nodes can reache each other through a path consisting of nodes only in this cluster.

DBSCAN can be seen as an application of DFS. However, DBSCAN differs from standard DFS from its useage of DFS-VISIT. In standard DFS, the algorithm goes deeper by calling DFS-VISIT($G, v$) on all unvisited children of node $u$. In DBSCAN, however, the algorithm goes deeper if and only if node $u$ satisfies extra conditions, which are specified by two user given value $Eps$ and $MinPts$. These extra conditions make the component generated from DBSCAN a meaningful cluster viewing from the density side. The pseudocode is shown below.

```
DBSCAN(S, Eps, MinPts)
 1  ClusterId = 1
 2  for each point u ∈ S
 3       if u.label ≠ NIL
 4            continue
 5       u.neighbor = REGION-QUERY(u, Eps)
 6       if ‖u.neighbor‖ < MinPts
 7            u.label = NOISE
 8       else
 9            EXPAND-CLUSTER(u, ClusterId, Eps, MinPts)
10            ClusterId = ClusterId + 1

EXPAND-CLUSTER(u, ClusterId, Eps, MinPts)
1  u.label = ClusterId
2  for each point v ∈ u.neighbor
3       if v.label == NIL
4            v.label = ClusterId
5            v.neighbor = REGION-QUERY(v, Eps)
6            if ‖v.neighbor‖ < MinPts
7                 EXPAND-CLUSTER(v, ClusterId, Eps, MinPts)
8       elseif v.label == NOISE
9            v.label = ClusterId
```

Similiar with DFS traverse, DBSCAN clustering is also divided into two procedures, DBSCAN and EXPAND-CLUSTER. DBSCAN functions as a wrapper function in the same way as DFS. This procedure goes through each point in the data set $S$. If the current point $u$ has been visited, the procedure skips it. Otherwise, further process continues. However, unlike calling DFS-VISIT on every unvisited $u$ unconditionaly as DFS does, in DBSCAN, another procedure EXPAND-CLUSTER is called if and only if $u$ has sufficient number of neighbors in a given range $Eps$. REGION-QUERY$(u, Eps)$ returns all the neighbors of $u$ with a distance no further than $Eps$. This is the extra condition mentioned earlier. Similiarly, this condition is also checked in EXPLAND-CLUSTER at line 6, as opposed to DFS-VISIT. Besides these two places, the rest of the two algorithms are the same. Details of extra conditions are fully explained in next section.

### 3.2.2 Notions of Density in Clusters

From the view of density, points in a space can be classified into three categories: core points, border points, and outliers/noise. The classification

criterion on a point $u$ bases on the size of its *Eps-neighborhood*, denoted as $N(u; Eps)$. The $N(u; Eps)$ represents the collection of all points whose distances to $u$ are within $Eps$, which is specified by user. More formally, $N(u; Eps) = \{v \mid dist(u, v) < Eps\}$.

Based on above definition, a point $u$ is a core point if and only if $\|N(u; Eps)\| \geq MinPts$, where both $Eps$ and $MinPts$ are user specified. After defining core points, it is relatively easy to define the other two categories. A border point $v$ is a neighbor point of a core point $u$, but $v$ is not a core point itself. Formally, $\|N(u; Eps)\| < MinPts$, and $v \in N(u; Eps)$, where $\|N(u; Eps)\| \geq MinPts$. For a outlier, it's a point $v$ which is neither a core point itself nor a neighbor point of a core point. An example graph is showing in Fig 3.3.

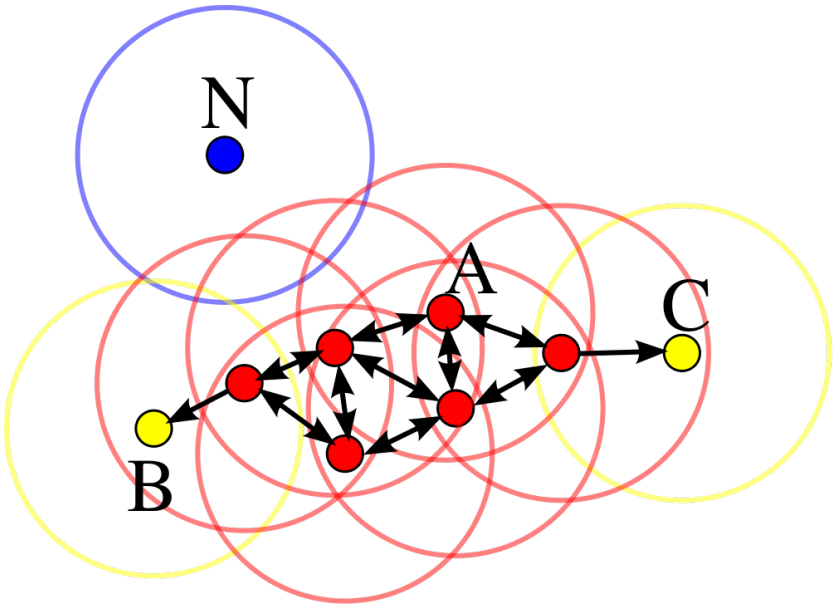



Figure 3.3: Illustration of core points(red), border points(yellow), and outlier(blue) [14]. In the illustration, $MinPts = 4$, $Eps$ is the radius of those circles.

As show in the figure, red points are core points since in each red circle, there are at least 4 points(including the center point). While for the yellow point, they don't have enough points in their circles, but they embed in one of the red circles. Thus, they are border points. As for the blue point, it has neither enough point in the blue cirle nor is in any red circle. Thus, the blue point is an outlier/noise.

### 3.2.3 Algorithm of DBSCAN

When you use `pdflatex` to render your thesis, you can include PDF images directly, as shown by Figure 3.4 below.

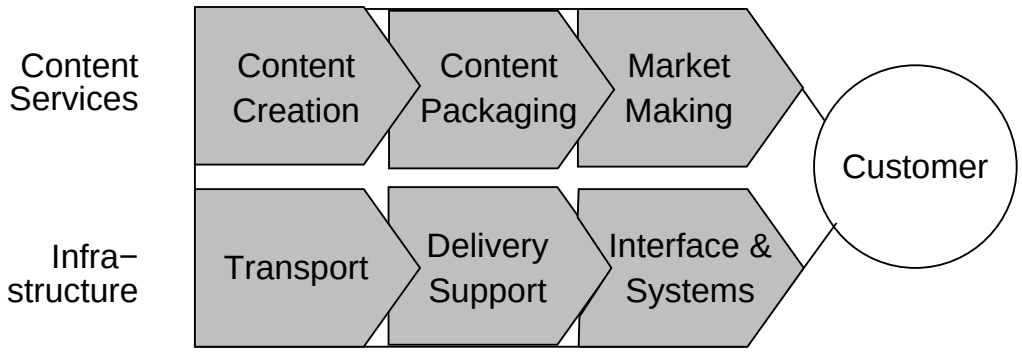



Figure 3.4: The INDICA two-layered value chain model.

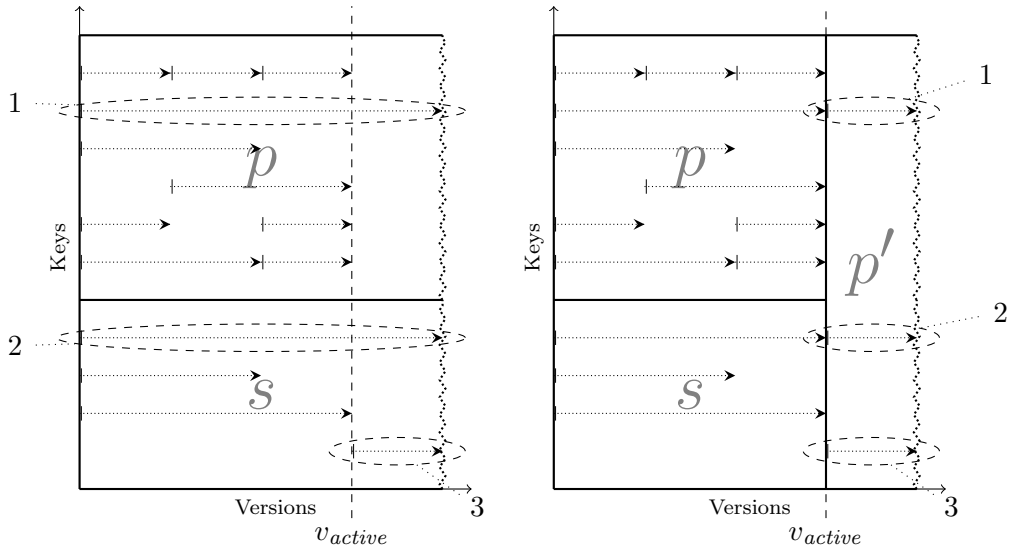
You can also include JPEG or PNG files, as shown by Figure 3.5.

Figure 3.5: Eeyore, or Ihaa, a very sad donkey.

You can create PDF files out of practically anything. In Windows, you can download PrimoPDF or CutePDF (or some such) and install a printing driver so that you can print directly to PDF files from any application. There are also tools that allow you to upload documents in common file formats and convert them to the PDF format. If you have PS or EPS files, you can use the tools `ps2pdf` or `epspdf` to convert your PS and EPS files to PDF.

Furthermore, most newer editor programs allow you to save directly to the PDF format. For vector editing, you could try Inkscape, which is a new open source WYSIWYG vector editor that allows you to save directly to PDF. For graphs, either export/print your graphs from OpenOffice Calc/Microsoft Excel to PDF format, and then add them; or use `gnuplot`, which can create PDF files directly (at least the new versions can). The terminal type is *pdf*, so the first line of your plot file should be something like `set term pdf ...`.

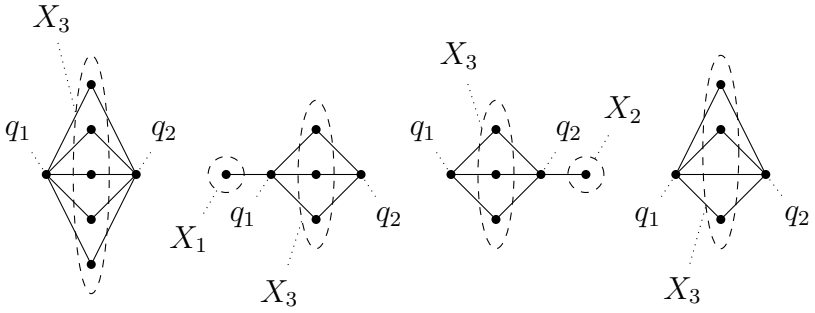
To get the most professional-looking graphics, you can encode them using the TikZ package (TikZ is a frontend for the PGF graphics formatting system). You can create practically any kind of technical images with TikZ, but it has a rather steep learning curve. Locate the manual (`pgfmanual.pdf`) from your LaTeX distribution and check it out. An example of TikZ-generated graphics is shown in Figure 3.6.



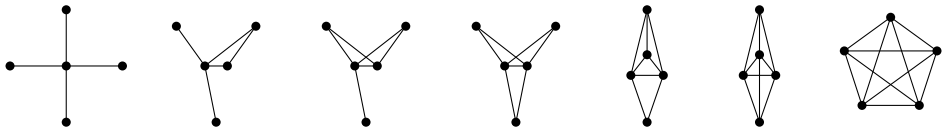

(a) Before (b) After

Figure 3.6: Example of a multiversion database page merge. This figure has been taken from the PhD thesis of Haapasalo [10].

Another example of graphics created with TikZ is shown in Figure 3.7. These show how graphs can be drawn and labeled. You can consult the example images and the PGF manual for more examples of what kinds figures you can draw with TikZ.




(a) Examples of obstruction graphs for the Ferry Problem



(b) Examples of star graphs

Figure 3.7: Examples of graphs draw with TikZ. These figures have been taken from a course report for the graph theory course [9].

# Chapter 4

# Methods

You have now stated your problem, and you are ready to do something about it! *How* are you going to do that? What methods do you use? You also need to review existing literature to justify your choices, meaning that why you have chosen the method to be applied in your work.

If you have not yet done any (real) metholodogical courses (but chosen introduction courses of different areas that are listed in the methodological courses list), now is the time to do so or at least check through material of suitable methodological courses. Good methodologial courses that consentrates especially to methods are presented in Table 4.1. Remember to explain the content of the tables (as with figures). In the table, the last column gives the research area where the methods are often used. Here we used table to give an example of tables. Abbreviations and Acronyms is also a long table. The difference is that longtables can continue to next page.

| **Code** | **Name** | **Methods** | **Area** |
|---|---|---|---|
| T-110.6130 | Systems Engineering for Data Communications Software | Computer simulations, mathematical modeling, experimental research, data analysis, and network service business research methods, (agile method) | T-110 |
| Mat-2.3170 Simulation (here is an example of multicolumn for tables) | | Details of how to build simulations | T-110 |
| S-38.3184 | Network Traffic Measurements and Analysis | How to measure and analyse network traffic | T-110 |

Table 4.1: Research methodology courses

# Chapter 5

# Implementation

You have now explained how you are going to tackle your problem. Go do that now! Come back when the problem is solved!

Now, how did you solve the problem? Explain how you implemented your solution, be it a software component, a custom-made FPGA, a fried jelly bean, or whatever. Describe the problems you encountered with your implementation work.

# Chapter 6

# Evaluation

You have done your work, but that's[1] not enough.

You also need to evaluate how well your implementation works. The nature of the evaluation depends on your problem, your method, and your implementation that are all described in the thesis before this chapter. If you have created a program for exact-text matching, then you measure how long it takes for your implementation to search for different patterns, and compare it against the implementation that was used before. If you have designed a process for managing software projects, you perhaps interview people working with a waterfall-style management process, have them adapt your management process, and interview them again after they have worked with your process for some time. See what's changed.

The important thing is that you can evaluate your success somehow. Remember that you do not have to succeed in making something spectacular; a total implementation failure may still give grounds for a very good master's thesis—if you can analyze what went wrong and what should have been done.

[1] By the way, do *not* use shorthands like this in your text! It is not professional! Always write out all the words: "that is".

# Chapter 7

# Discussion

At this point, you will have some insightful thoughts on your implementation and you may have ideas on what could be done in the future. This chapter is a good place to discuss your thesis as a whole and to show your professor that you have really understood some non-trivial aspects of the methods you used...

# Chapter 8

# Conclusions

Time to wrap it up! Write down the most important findings from your work. Like the introduction, this chapter is not very long. Two to four pages might be a good limit.

# Bibliography


[1] Aalto University, Helsinki University of Technology, Committee on Academic Affairs. Guidelines for master's thesis evaluation, December 4 2008. http://www.tkk.fi/fi/opinnot/opintohallinto/paatokset/guidelinemastersthesisevaluation.pdf (in English), http://www.tkk.fi/fi/opinnot/opintohallinto/paatokset/diplomityoarviointiohje.pdf (in Finnish).

[2] Aalto University Library. webpage, March 2 2011. How to find - brief guide to finding publications and information. Accessed 4.3.2011.

[3] Aalto University Library. Making a bibliography. webpage, March 2 2011. http://otalib.aalto.fi/en/instructions/guides/citations/bibliography/. Accessed 4.2.2011.

[4] Aalto-yliopisto, Degree programme of Computer Science and Engineering. Instruction pages: Diplomityö, 2010. http://information.tkk.fi/fi/opinnot/tietotekniikka/diplomityo/ (in Finnish), http://information.tkk.fi/en/studies/cse/thesis/ (in English). Accessed 9.2.2011.

[5] Aloise, D., Deshpande, A., Hansen, P., and Popat, P. Np-hardness of euclidean sum-of-squares clustering. *Machine learning 75*, 2 (2009), 245–248.

[6] Becker, B., Gschwind, S., Ohler, T., Seeger, B., and Widmayer, P. An asymptotically optimal multiversion B-tree. *The VLDB Journal 5*, 4 (1996), 264–275.

[7] Campello, R. J., Moulavi, D., Zimek, A., and Sander, J. Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Transactions on Knowledge Discovery from Data (TKDD) 10*, 1 (2015), 5.

[8] Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd* (1996), vol. 96, pp. 226–231.

[9] Göös, M., Haapasalo, T., and Moisio, L. Finding hard instances of the ferry problem. Course report for the course T-79.5203/S-72.2420 Graph theory, Aalto SCI, 2010.

[10] Haapasalo, T. *Accessing Multiversion Data in Database Transactions*. PhD thesis, Department of Computer Science and Engineering, Aalto University School of Science and Technology, Espoo, Finland, 2010. `http://lib.tkk.fi/Diss/2010/isbn9789526033600/`.

[11] Haapasalo, T., Jaluta, I., Seeger, B., Sippu, S., and Soisalon-Soininen, E. Transactions on the multiversion $B^+$-tree. In *Proceedings of the 12th Conference in Extending Database Technology* (2009), pp. 1064–1075.

[12] He, Z., Xu, X., and Deng, S. Discovering cluster-based local outliers. *Pattern Recognition Letters 24*, 9 (2003), 1641–1650.

[13] Lloyd, S. Least squares quantization in pcm. *IEEE transactions on information theory 28*, 2 (1982), 129–137.

[14] Wikipedia. Dbscan — Wikipedia, the free encyclopedia, 2016. [Online: accessed 09-August-2016].

# Appendix A

# First appendix

This is the first appendix. You could put some test images or verbose data in an appendix, if there is too much data to fit in the actual text nicely.

For now, the Aalto logo variants are shown in Figure A.1.


(a) In English


(b) Suomeksi


(c) På svenska

Figure A.1: Aalto logo variants