proceedings of the eight StudCol 2011, 20 and 21 April, 2011 Groningen

Conference Paper · April 2011

CITATIONS
0

READS
1,438

3 authors, including:

Rein Smedinga
University of Groningen
30 PUBLICATIONS **109** CITATIONS
SEE PROFILE

Michael Biehl
University of Groningen
336 PUBLICATIONS **7,671** CITATIONS
SEE PROFILE

SC@RUG 2011 proceedings

Rein Smedinga
Michael Biehl
Femke Kramer
editors

2011
Groningen

ISBN: 978-90-367-5019-6
Publisher: Bibliotheek der R.U.
Title: Proceedings 8th Student Colloquium 2010-2011
Computing Science, University of Groningen
NUR-code: 980

About SC@RUG

Introduction SC@RUG (or student colloquium in full) is a course that master students in computing science follow in the first year of their master study at the University of Groningen.

In the academic year 2010-2011 SC@RUG was organized as a conference for the eigth time. Students wrote a paper, participated in the review process, gave a presentation and were session chair during the conference.

The organizers Rein Smedinga, Michael Biehl and Femke Kramer would like to thank all colleagues, who cooperated in this SC@RUG by collecting sets of papers to be used by the students and by being an expert reviewer during the review process. They also would like to thank Janneke Geertsema for her workshops on presentation techniques and speech therapy.

Organizational matters SC@RUG 2011 was organized as follows. Students were expected to work in teams of two. The student teams could choose between different sets of papers, that were made available through *Nestor*, the digital learning environment of the university. Each set of papers consisted of about three papers about the same subject (within Computing Science). Some sets of papers contained conflicting opinions. Students were instructed to write a survey paper about this subject including the different approaches in the given papers. The paper should compare the theory in each of the papers in the set and include own conclusions about the subject.
Eight teams proposed their own subject.

After submission their papers, each student was assigned one paper to review using a standard review form. The staff member who had provided the set of papers was also asked to fill in such a form. Thus, each paper was reviewed three times (twice by peer reviewers and once by the expert reviewer). Each review form was made available to the authors of the paper through *Nestor*.

All papers could be rewritten and resubmitted, independent of the conclusions from the review. After resubmission each reviewer was asked to re-review the same paper and to conclude whether the paper had improved. Rereviewers could accept or reject a paper. All accepted papers can be found in these proceedings.

All students were asked to present their paper at the conference and act as a chair and discussion leader during one of the other presentations. Half of the participants were asked to organize one day of the conference day (i.e., to make the time tables, invite people, look for sponsoring, etc.) The audience graded both the presentation and the chairing and leading the discussion.

In her lectures about communication in science, Femke Kramer explained how conferences work, how researchers review each other's papers, and how they communicate their findings with a compelling storyline and cleverly designed images.

Michael Biehl taught a workshop on writing a scientific paper and Janneke Geertsema gave workshops on presentation techniques and speech therapy that were very well appreciated by the participants.

Rein Smedinga did the overall coordination, administration and served as the main manager of *Nestor*.

Students were graded on the writing process, the review process and on the presentation. Writing and rewriting counted for 50% (here we used the grades given by the reviewers and the re-reviewers), the review process itself for 15% and the presentation for 35% (including 5% for the grading of being a chair or discussion leader during the conference). For the grading of the presentations we used the judgments from the audience and calculated the average of these.

In this edition of SC@RUG students were videotaped during their presentation. The recordings were published on *Nestor* for self reflection.

On 20 and 21 April 2011, the actual conference took place. Each paper was presented by both authors. On Wednesday 7 presentations were given and on Thursday 15, each consisting of a total of 15 minutes for the introduction, the actual presentation and the discussion. On Wednesday Louwarnoud van der Duim from the UOCG was the keynote speaker and on Thursday Muhittin Hasancioglu, General Manager IT & IM Technology from Shell, gave a key note presentation.

As mentioned before, each presenter also had to act as a chair and discussion leader for another presentation during that day. The audience was asked to fill in a questionnaire and grade the presentations, the chairing and leading the discussion. Participants not selected as chair were asked to organize the day.

Thanks We could not have achieved the ambitious goal of this course without the invaluable help of the following expert reviewers: Henk Bekker, Frank Brokken, Pavel Bulanov, Tobias Isenberg, Jan Jongejan, Alexander Lazovik, Andrea Pagani, Nikolai Petkov, Alexandru Telea, Dan Tofan, Marco Wiering, and Michael Wilkinson

Also, the organizers would like to thank the *School of Computing and Cognition* for making it possible to publish these proceedings and sponsoring the conference.

Rein Smedinga
Michael Biehl
Femke Kramer

Contents

Detecting edges using the Marr-Hildreth approach

Rudy Schoenmaker and Jeroen de Groot

Abstract— The Marr-Hildreth edge detection method operates by convolving the image with the Laplacian of the Gaussian function, or, as a fast approximation by Difference of Gaussians. [6] In the next step zero-crossings are detected in the filtered result to obtain the edges. Therefore it is possible to detect edges of visual objects within natural images, these edges are necessary for further processing, like object or contour detection. Several applications analyze visual data without human interaction, these applications have to recognize objects out of the visual data in order to process the relevant data. Edge detection is an important task when we want to recognize objects without human interaction. It is one of the first steps that has to be taken in order to get a better understanding about the image or video capture for further processing. We implemented the Marr-Hildreth algorithm to detect edges and compared the results of the Marr-Hildreth algorithm with the Berkeley Segmentation Dataset. This is a dataset that contains images segmented by human interaction [5]. Through this dataset we are able to determine the performance of the Marr-Hildreth algorithm. The results are not satisfying compared to other edge detectors but this is explained by the human interpretation of objects, the main boundaries around objects are detected. However a lot of noise is still left in the image, other techniques are needed tot remove this kind of data. We did not implemented such methods, because we are mainly interested in the performance of the Marr-Hildreth algorithm itself.

Index Terms—Marr-Hildreth, Berkeley Segmentation Dataset, edge detection, image processing, zero-crossing, Laplacian, Gaussian

1 Introduction

Edge detection is frequently used in image processing and analysis. Detecting edges in an image is an important step towards creating a better overview of the data available within the image especially in respect to the visual objects in that image. With this data we are a step closer to recognize objects within the data. Many practical applications use edge detection algorithms to create contours for object recognition, for example robot vision or medical image analyses.

There is no specific mathematical definition of what a contour is [9]. We as humans are briefly trained from when we are born to recognize objects within images and the real world. Since computers do not have the ability to distinguish objects in a given natural image, we will propose a solution to this problem by detecting edges using the Marr-Hildreth algorithm. These edges will form the contours of the objects within the image. We declare contours as *the set of lines that human observers would consent on to be the contours in that image* [1] The process of detecting edges is non-trivial for a computer, since objects and background may consist of multiple variance in the gradient, making it for the computer difficult to make a clear distinction between the object and the background. Also when an image is not sharp we can encounter problems in detecting the edges, since the background fades into the object, which makes detection more difficult.

There are several algorithms available for edge detection, like Sobel [10], Prewitt [8] or the Canny edge detector [7]. Most of them involve differentiation or gradient measurements. The Canny edge detector is a good working detector but does not create closed contours, in which we are interested. The Marr-Hildreth edge detector is known for his ability to create closed contours. We have to be careful with this, because the Marr-Hildreth edge detector is also know as an detector which marks edges which in fact are not really there, also known as ghost edges. As mentioned before we are interested in the edge data to have the possibility to recognize objects in the image. Therefore we have a need to generate binary data (black = 0, white = 1) which represents the edges (marked with 1) from the images. With this data further processing can be done, like actually classifying objects.

In this paper we will give a brief description of the Marr-Hildreth edge detection algorithm, which is explained in section 2.1 and how well it performs is described in section 3. We measure the performance of the algorithm by comparing our results to the Berkeley Segmentation DataSet. This is a set of images where the edges are manually drawn by human interaction, this is more briefly described in section 2.2. In section 3 we will present the results of our research and finally a discussion is provided in section 4.

- *Rudy Schoenmaker, s1938150, E-mail: rudyschoenmaker@live.com*
- *Jeroen de Groot, s1921320, E-mail: jrdegroot@gmail.com*

2 Methods and Materials

In this section the Marr-Hildreth algorithm for detecting edges and the Berkeley Segmentation set are explained. In the first subsection 2.1 we give a brief description about the Marr-Hildreth algorithm, especially how it works and how it differs from other edge detecting algorithms. Why are these differences interesting for us? In subsection 2.2 the Berkeley Segmentation Dataset is explained, we explain why and how we use this dataset to verify our results obtained with the Marr-Hildreth algorithm.

2.1 Marr-Hildreth Algorithm

The Marr-Hildreth method is a gradient based operator which uses the Laplacian to take the second derivative of an image. The idea is that if there is a step difference in the intensity of the image, it will be represented in the second derivative by a zero crossing as shown in Fig: 3. For calculating these differences in intensity we apply a two dimensional Laplacian to the image. This operation is the equivalent of taking the second derivative of the image, also known as the zero-crossings of the Laplacian. The definition of the Laplacian operator is presented in the following equation [2]:

$$G(x,y) = F(x,y) = \frac{\delta^2 f}{\delta x^2} + \frac{\delta^2 f}{\delta y^2} = 0 \tag{1}$$

As can be seen the Laplacian operator produces a non-directional second derivative [G(x,y)] of a two dimensional image [F(x,y)]. The locus of zeros in the resulting function is the locus of the minima or maxima in the gradient or else the gradient itself is zero. The loci of zero-crossings are the places where the change in gradient goes from positive to negative or vice versa. These are the inflections of the images and such inflections occur at edges of objects in the image.

As mentioned before the edges are detected by looking at the zero-crossings of the second derivative by applying the Laplacian, to understand the zero-crossing we take the second derivative of the image and plot it in Fig 3. Where the slope crosses the origin the gradient changes from positive to negative or vice versa and therefore the possibility of hitting an edge on an object is high and we mark it as an edge pixel.

To implement the Laplacian operation we have to use digital filtering which is a replacement of each pixel in the image with a weighted sum of its neighbors, the array of weights is called a 'kernel'. To implement the Laplacian operation we have to use a 3 x 3 kernel as shown

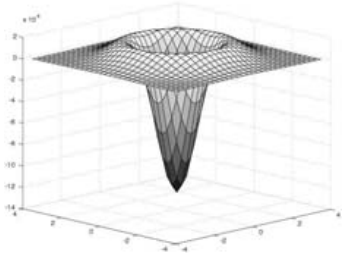


Fig. 1: A 3d graphical representation of the The Laplacian matrix as can be seen the "Mexican Hat" shape is very clear

0	-1	0
-1	4	-1
0	-1	0

Table 1: Commonly used discrete approximations to the Laplacian filter

in Fig 1. This kernel replaces every pixel by the second difference of the image at that certain point in the x and y directions.

Differentiation often generates spurious edges due to the presence of frequency noise, therefore we smooth the image to get better results. Smoothing will be done by a kernel which is in most cases larger than 3 x 3. The ideal size is as large as the lowest spatial frequency that is considered noise, this will be different for every image. Marr and Hildreth use the Gaussian kernel to smooth the image. They adjusted the amount of smoothing by the size of the array, which is proportional to the standard deviation of the Gaussian. Since the second order difference and the Gaussian smoothing operation are both linear, Marr and Hildreth combined those into the Laplacian of a Gaussian. When this function is plotted it has the shape of a "Mexican hat" as shown in Fig 1. [3]

Why the Marr-Hildreth algorithm?

The Marr-Hildreth algorithm creates closed contours as can be seen in Fig 2. This is very important for detecting contours or objects. Since we do not have to deal with edge reconstruction, the object classification can be done more easily. Nevertheless we have a lot of noise as can be seen within the wheel, which again makes the classification more difficult. Therefore the Marr-Hildreth is also not the perfect edge detector, but is the most interesting in our study case, because Prewitt, Sobel and Canny edge do not create closed contours.

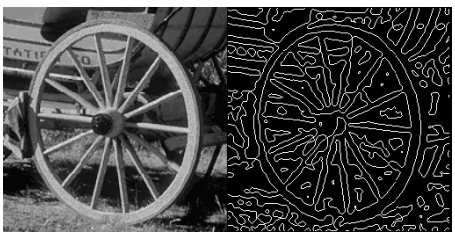


Fig. 2: A natural image of a wheel, and a the result from the Marr-Hildreth algorithm. The closed contours are very clear in this example.

2.2 Berkeley Segmentation Dataset

The Berkeley Segmentation Dataset is a dataset which consists of sets of images which are segmented by human interaction. There are 300 image photographs, which all have roughly 5 segmented images done by different human beings. There is a set of segmented images available for the colored picture as well as for a gray-level version of the picture. Segmentation is done by human hand, therefore it can consist of some localization error as well as the interpretation of boundaries differing per individual. Therefore the segmented image sets can differ a little, but besides the level of detail for the contours most of the segmented image are consistent. As stated on their website the dataset is provided for an empirical basis for scientific research on image segmentation and contour detection [5]. They have used this data to develop new boundary detection algorithms. On the website several algorithms are discussed, like the Global Probability of Boundary, Xren and Texture gradient [5]. In this research for edge detection with the Marr-Hildreth algorithm we have used the Berkeley Segmentation Dataset to provide a visual performance measure for the Marr-Hildreth algorithm compared to the segmented images created by human interaction.

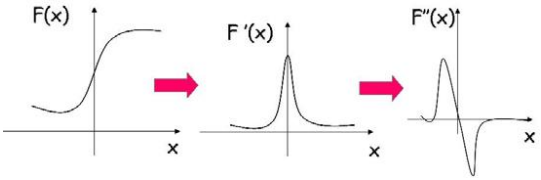


Fig. 3: Graphical representation of the zero-crossing. The first slope is the Laplacian function plotted on an x and y axis, followed by the 1st and 2nd derivative of the function. As can be seen the zero-crossing is very clear, because the slope crosses the x and y axis at the origin.

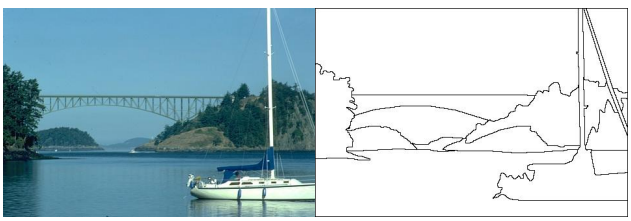


Fig. 4: An example image from the Berkely Segmentation Dataset and an segmented image. The segmentation shown in the right part is achieved by human interaction

3 Results

The aim of this evaluation is to provide a better understanding of the performance of the Marr-Hildreth algorithm, and how well it performs compared to images segmented by human beings. The Berkeley Segmentation Dataset (BSDS) provides a good measure for this scientific research. Having the human segmented images compared to the Contoured image trough the Marr-Hildreth algorithm, we can see if the algorithm creates correct lines and edges.

In this evaluation an original image from the Berkeley Segmentation Dataset is taken and the Marr-Hildreth algorithm is applied on this image to detect the edges. The result obtained is compared with the segmented image done by humans, as can be seen as a graphical view in Fig 5. By this comparison we will see how well the Marr-Hildreth algorithm performs. The BSDS contains images from several levels of difficulty. Some images have a clear distinction between background and objects, while other images have objects which fade away in the background. In order to get a good evaluation we compared the algorithm with multiple images from several levels of difficulty.

For measuring the performance of the Marr-Hildreth algorithm in comparison with the Berkeley Segmentation Dataset we used the following algorithm [4]:

$$P = \frac{t_p}{t_p + f_p + f_n} \tag{2}$$

in which f_p and f_n are the false positives and false negative pixels detected. P gets an accurate value between 0 and 1. t_p is the number of correctly detected contour pixels. When P has the value of 1, this

means that all the contour pixels are correctly detected and there are no false detected contour pixels. While in a perfect world this algorithm would be sufficient, and only the level of detail from the interpretation of a human would differ. Humans make errors, by not drawing the edges perfectly around and object, or they have a different interpretation of objects. In order to compensate for this localization error a window-based approach is used. A pixel is also counted as a correct one when it lies in the 5x5 window size.

Fig. 5: A graphical representation of the overlap from the result from the Marr-Hildreth algorithm on the bridge image and BSDS segmented image. The white lines represented the edges obtained by the Marr-Hildreth algorithm, and the yellow line represent the contours created with human interaction.

Fig. 6: The original and the result from the Marr-Hildreth algorithm from the image of a bird

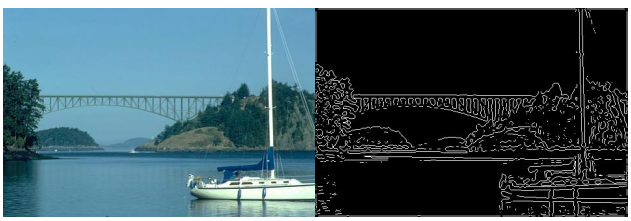


Fig. 7: The original and the result from the Marr-Hildreth algorithm from the image of a bridge

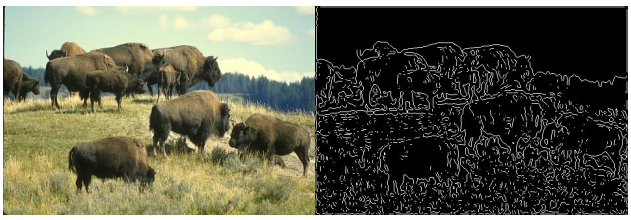


Fig. 8: The original and the result from the Marr-Hildreth algorithm from the image of some buffalos

Fig. 6 contains the original and the segmented image by the Marr-Hildreth algorithm. This image has a clear distinction between objects and the background. Which would make it more easy for a computer to create the contours of the bird. When looking at the segmented image you can see that the contours are indeed easily spotted around the bird, as well as around the tree itself.

Fig. 7 contains the original image from a bridge, as well as the segmented image by the Marr-Hildreth algorithm. This image is compared to the bird image more difficult to contour. The distinction between the bridge the water and the boat are easily noticeable, but the level of detail in the objects itself makes it harder for the computer to only contour the object. Even when looking at the segmented images in the BSDS we can see that humans contour the image differently in the level of detail [4]. The segmented image by the Marr-Hildreth algorithm does separate the bridge and boat from the background nicely, but on the other hand it detects more edges inside objects then the ground truth.

Fig. 8 contains the original image from buffalos in a landscape. This image does not have a clear distinction between the objects(animals) and the background(landscape). The objects fade away in the background. This will make it very difficult for the algorithm to separate objects from the background. When looking at the result from the Marr-Hildreth algorithm we see that the algorithm has trouble separating the objects from background. Looking at the picture of the beast, the human contoured image only draws contours on the animal itself. While the Marr-Hildreth algorithm has difficulties in the grass area, not knowing whether the small edges in the grass should be accounted as contours. While a human subject can easily distinguish the relevant animals from the background which we are not interested in.

The Berkeley Segmentation DataSet has around 5 segmented images for each photograph. In order to compensate for the human localization error and give a more realistic view of the performance of the Marr-Hildreth the image is convolved against all the 5 images and an average is calculated.

Image	Min performance	Max Performance
Bird.jpg	0.34	0.55
Bridge.jpg	0.31	0.45
Animals.jpg	0.20	0.41

"When you can measure what you are speaking about and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of the meager and unsatisfactory kind." - Lord Kelvin

Using the performance measure we explained, we created this table with the values on the accuracy of the segmented images. Looking at this table we can validate our visual analyses of the images. The image of the bird has the best result, while the image of the animals performs a lot worse. If we visually compare the images with the several other algorithms in the BSDS. the Marr-Hildreth algorithm is not the best algorithm for detecting edges.

The Berkeley Segmented dataset uses a different performance measure as we used in our research. They use a precision-recall curve algorithm which is more explained into detail on their site [5]. Because they use a different performance measure we cannot compare the actual exact numbers. Therefore we compared the results of our research with the BSDS visually. We did compare other algorithms like the Canny edge detection or Prewitt's and Sobel's masks using our own performance measure. From which the results are posted in the table below.

Image	MH	Canny	Sobel	Prewitt
Bird.jpg	0.49	0.62	0.81	0.81
Bridge.jpg	0.40	0.37	0.46	0.48
Animals.jpg	0.39	0.34	0.49	0.49

Looking at the results, we can see that the Marr-Hildreth algorithm performs poorly compared to the Sobel and Prewitt mask on the first picture. While the results of the canny edge detector are almost equivalent. This can be explained by the difference in detail between the ground truth, and the segmented image. The Marr-Hildreth as well as the canny edge detectors both keep a certain level of detail. While the

Sobel and Prewitt leave out as much as detail as possible with the risk of losing relevant data. This means that there are less false positive pixels making their performance number go up.

4 Discussion

We compared the algorithm with the Sobel Prewitt and Canny edge detection algorithms using the performance measure explained in section 3. The performance of the Marr-Hildreth algorithm was not that satisfactory compared to the Sobel and Prewitt masks, the Canny edge detector gave almost the same results. While the Canny and Marr-Hildreth edge detection methods are known to be better edge detectors, the results can be explained through human interpretation of objects. The Canny and Marr-Hildreth algorithms create a way more detailed edge spectrum inside objects, mostly useful small details are marked as edges. While on the human segmented images only the outside of the objects are marked. The Sobel and Prewitt are fast and also leave out high level of detail (small details are not recognized). This explains why their performance looks better when we just look at the numbers. When we take a look at the actual segmented images we see that the Canny and Marr-Hildreth are actually more accurate in detecting edges.

When the background gets a higher variance in the gradient, the Marr-Hildreth algorithm has troubles leaving the irrelevant edges out of the contours and marks it as objects. Almost all the relevant contours which are detected by the human subjects are also detected by the Marr-Hildreth algorithm. As already discussed in the introduction the Marr-Hildreth algorithm is known to create ghost edges. On the images of the Berkeley segmentation database this only appeared on the most difficult to analyze images.

Looking at the segmented images of the algorithms the Berkeley Segmentation Dataset already provided, the Marr-Hildreth Algorithm is not the best algorithm available. There are some better performing algorithms. Which comes of no surprise since the Marr-Hildreth algorithm is an outdated algorithm.

References

[1] N. P. Giuseppe Papari . Edge and line oriented contour detection: State of the art. *Imavis*, Elsevier(03023):25, 2009.

[2] T. S. Jr. Edge detection in images using marr-hildreth filtering techniques. *Neuroscience Methods*, Elsevier(26):75–82, June 1988.

[3] E. Nadernejad. Edge detection techniques: Evaluations and comparisons. *Applied Mathematical Sciences*, 31(2):1507–1520, 2008.

[4] M. A. Nicolai Petkov. Contour detection based on nonclassical receptive field inhibition. *Transactions On Image Proccesing*, 12(7):11, 2003.

[5] C. F. Pablo Arbelaez and D. Martin. The berkeley segmentation dataset and benchmark. http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/, 2007.

[6] Wikipedia. Marr-hildreth algorithm. http://en.wikipedia.org/wiki/Marr-Hildrethalgorithm, 2010.

[7] Wikipedia. Canny edge detector. http://en.wikipedia.org/wiki/Cannyedgedetector, 2011.

[8] Wikipedia. Prewitt operator. http://en.wikipedia.org/wiki/Prewittoperator, 2011.

[9] Wikipedia. Shilouette. http://en.wikipedia.org/wiki/Silhouette, 2011.

[10] Wikipedia. Sobel operator. http://en.wikipedia.org/wiki/Sobeloperator, 2011.

Contour detection: Removing spurious edges

Pieter Stavast , Sander Kelders

Abstract— We propose two post processing steps after contour detection for spurious edge removal. One is using connected components the other is based on line orientation and morphological filtering. We construct two contour detectors on which we will test our spurious edge removal techniques. A Canny edge detector and a Nonclassical Receptive Field Inhibition based contour detector. Spurious edge removal through connected components seems to work better than using linefilters. Edge removal combined with the Canny edge detector works not as well as with Nonclassical Receptive Field Inhibition. The results we present here show promise but more research is needed to reduce enough of the spurious edges without removing the contour edges.

Index Terms—Contour detection, spurious edge removal, texture inhibition, morphological filters, connected components

1 Introduction

Contour detection is an important topic of research because it expands the possibilities of automated processes. Many automated processes use camera's to collect information about their surroundings, and contour detection plays a vital role in finding objects of interest. Many contour detection methods are available today such as the Watershed method [2] or contour detection based on the Canny Edge Detector[2]. Most of the known detectors work well with artificial images or images from a controlled environment, but they fall short when faced with natural images which contain a lot of texture.

The edge detectors respond to the texture edges, but since these are not usually part of a contour we are not interested in them. Since the human visual system is very good at detecting objects, it has been an inspiration to researchers, and contour detection methods based on the human visual system have been developed. Contour detection using Nonclassical Receptive Field Inhibition [3] (NRFI) is based on the observation that the human visual system inhibits texture edges. The method detects contours by finding all the edges, and depending on the presence of edges in its surrounding suppresses them. Through this suppression better results are achieved in natural images than most detectors which are not biologically motivated. However, even with the improved results many texture induced edges remain. These unwanted edges are the Spurious Edges we want to eliminate.

To finalize the contour detector we propose another step in which the spurious edges are eliminated. In this paper we describe two methods: one based on morphological line filters and one based on connected components. We also use a Canny Edge Detector to compare the usefulness of Spurious Edge Removal (SER) using non biologically motivated contour detectors to SER with biologically motivated contour detectors.

In this paper we first describe the contour detectors we use. Next we discuss the SER in detail and lastly we show and discuss the experiments and results.

2 Contour detectors

Before we can apply SER we need to have credible test images. To this end we implement two contour detectors. A contour detector based on the Canny Edge Detector and a contour detector based on Nonclassical Receptive Field Inhibition.

2.1 Canny Edge Detection

The Canny edge detection algorithm was designed with three objectives in mind. It should have a low error rate, the true edges should be found and there should not be any spurious edges. The second objective is that edges should be as close as possible to the true edges.

- *Pieter Stavast is a Computing Science Master student at the Rijksunversiteit Groningen.*
- *Sander Kelders is a Computing Science Master student at the Rijksunversiteit Groningen.*

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Table 1: Sobel filters

The last objective is only marking one point for each edge found. Our implementation is build as described in *Digital Image Processing* [2].

Spurious edges in images can be caused by noise or texture. Canny edge detection uses a Gaussian filter to reduce the influence of texture and noise. After Gaussian filtering, Canny edge detection uses a filter to detect horizontal, vertical and diagonal edges. Our implementation uses a Sobel filter because Sobel masks have better noise-suppression then for example a Prewitt or Roberts mask[2]. Sobel uses two masks to calculate the first derivative in horizontal and vertical direction. The masks are shown in Table 1. The Gaussian and Sobel masks can be combined by convolving them with each other. So to obtain the the final filter we convolve the Gaussian filter with both Sobel masks.

After filtering the image with the combined filter you get the derivatives in two directions. The total edge gradient is calculated using Equation 1.

$$G = \sqrt{G_x^2 + G_y^2} \tag{1}$$

When edges are extracted directly after this step the edges are wide around the maxima because of the Gaussian filtering. To thin the edges nonmaxima suppression is used. Nonmaxima suppression sets the gradient value of a pixel to 0 if one of its neighbours along the same direction is higher. The direction of an edge is calculated using Equation 2. From the angles vertical, horizontal, forward diagonal and backward diagonal edges are extracted.

$$\arctan(G_y/G_x) \tag{2}$$

After the nonmaxima suppression thresholding is done to get a high and a low threshold. The high threshold is calculated by giving a percentage of pixels that should be above a certain edge value. The low value is calculated by dividing the high threshold by two. All the points above the high threshold are directly marked as edges. All the points above the low threshold that are connected to points above the high threshold are also marked as edges. The remaining pixels are marked as non edges.

2.2 Nonclassical Receptive Field Inhibition

The implementation of NRFI is based on the work of Grigorescu *et al* and we use the same filter and inhibition techniques. Here we will present a summary. For the full explanation we refer to *Contour Detection Based on Nonclassical Receptive Field Inhibition*[3].

In the human visual cortex, cells exist which respond to edges with a certain orientation. These cells can be modeled by a family of Gabor functions [4]. This family of functions in given by Equation (3).

$$g_{\lambda,\sigma,\theta,\varphi}(x,y) = e^{-\frac{\widetilde{x}^2+\gamma^2\widetilde{y}^2}{2\sigma^2}}\cos(2\pi\frac{\widetilde{x}}{\lambda}+\varphi)$$
$$\widetilde{x} = x\ \cos\ \theta\ +\ y\ \sin\ \theta$$
$$\widetilde{y} = -x\ \sin\ \theta\ +\ y\ \cos\ \theta \tag{3}$$

In Equation (3) γ determines the ellipticity. We fix this at $\gamma = 0.5$. The standard deviation of the Gaussian is determined by σ, and as such it determines the size of the receptive field. By varying σ we can determine the size of the edges our filter will respond to. We let σ depend on the size of the filter used. We use square filters of $N \times N$ pixels and set sigma to $\sigma = \frac{N}{10}$. The parameter θ determines the orientation of the filter. By varying θ we can rotate the filter, and let it respond to edges with a different orientation. The parameter λ determines the frequency of the cosine factor. Together with σ it determines the frequency of the filter and the number of inhibitory and excitatory zones. We fix this ratio at $\frac{\sigma}{\lambda} = 0.56$ so we have just over two periods in the receptive field. The parameter φ is a phase offset, which determines where in the ellipse the inhibitory and excitatory zones are. An example filter is given in Figure 1.

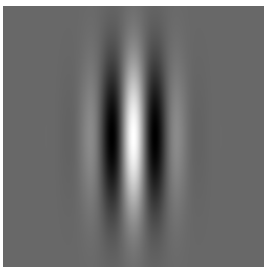


Fig. 1: The intensity map of an example Gabor Filter. With $\theta = 0$ the cell is vertically oriented and will respond to vertical edges. White areas contain values greater than zero, black areas contain values less than zero and gray areas contain values with negligible influence.

By convolving an image with a Gabor Filter $g_{\lambda,\sigma,\theta,\varphi}$ we get a result $r_{\lambda,\sigma,\theta,\varphi}$.

With the model of a simple cell we can now model more complex cells. We do this with the Gabor Energy model [5, 1, 6]. In this model the responses of a pair of simple cells with a phase difference of $\frac{\pi}{2}$ are combined according to Equation (4).

$$E_{\lambda,\sigma,\theta} = \sqrt{r^2_{\lambda,\sigma,\theta,0} + r^2_{\lambda,\sigma,\theta,-\frac{\pi}{2}}} \tag{4}$$

We will use Gabor Energy maps $E_{\lambda,\sigma,\theta_i}$ of different orientations θ_i given by Equation (5) and combine these with NRFI.

$$\theta_i = \frac{(i-1)\pi}{N_\theta}, i = 1,2,...,N_\theta \tag{5}$$

Now we construct an inhibition term. We recognize two sorts of inhibition. The first is anisotropic inhibition, in which only edge responses with the same orientation as a central response contribute to the suppression. The second is isotropic inhibition, in which all edge responses contribute to the suppression. To define the field which contributes to the suppression, we construct a weight function $w_\sigma(x,y)$ given by Equation (6). The difference of the Gaussians, Equation (8), constructs a ring around the Receptive Field and Equation (7) guarantees positive values in Equation (6).

$$w_\sigma(x,y) = \frac{1}{||H(D0G_\sigma)||_1} H(DoG_\sigma(x,y)) \tag{6}$$

$$H(z) = \begin{cases} 0 & if\ z \le 0 \\ z & if\ z \ge 0 \end{cases} \tag{7}$$

$$DoG - \sigma = \frac{1}{2\pi(4\sigma)^2} e^{-\frac{x^2+y^2}{2(4\sigma)^2}} - \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{8}$$

For both forms of inhibition we construct an inhibition term. For anisotropic inhibition this is defined in Equation (9) as a convolution of the Gabor Energy with the weight function.

$$t^A_{\lambda,\sigma,\theta_i}(x,y) = (E_{\lambda,\sigma,\theta_i} * w_\sigma)(x,y) \tag{9}$$

The result of the inhibition of one orientation is defined by Equation (10), and we will use this to determine the result in Equation (11). In both Equation (10) and Equation (11) α determines the impact of the inhibition. In our implementation we fix it at $\alpha = 1$.

$$\widetilde{b}^{A,\alpha}_{\lambda,\sigma,\theta_i}(x,y) = H(E_{\lambda,\sigma,\theta_i}(x,y) - \alpha t^A_{\lambda,\sigma,\theta_i}(x,y)) \tag{10}$$

$$b^{A,\alpha}_{\lambda,\sigma}(x,y) = max\{\widetilde{b}^{A,\alpha}_{\lambda,\sigma,\theta_i}(x,y)\ |i = 1,...,N_\theta\} \tag{11}$$

We also construct an orientation map $\Theta^A(x,y)$ with which $b^{A,\alpha}_{\lambda,\sigma}(x,y)$ was achieved in Equation (12). We will use this orientation map in the final steps of the contour detection.

$$\Theta^A(x,y) = \theta_k, k = arg\ max\{\widetilde{b}^{A,\alpha}_{\lambda,\sigma,\theta_i(x,y)} | i = 1,...,N_\theta\} \tag{12}$$

For isotropic inhibition we construct a term, which produces the maximum response of the Gabor Filter: Equation (13) and the orientation map, which we will use in the final steps of the contour detection: Equation (14).

$$\hat{E}_{\lambda,\sigma} = max\{E_{\lambda,\sigma,\theta_i}(x,y) | i = 1,...,N_\theta\} \tag{13}$$

$$\Theta^I(x,y) = \theta_k, k = arg\ max\{E_{\lambda,\sigma,\theta_i(x,y)} | i = 1,...,N_\theta\} \tag{14}$$

The inhibition term $t^I_{\lambda,\sigma}(x,y)$ is now computed by the convolution of the maximum response of the Gabor Filter with the weight function: Equation (15).

$$t^I_{\lambda,\sigma}(x,y) = (\hat{E}_{\lambda,\sigma} * w_\sigma(x,y)) \tag{15}$$

The contours are now determined by subtracting the inhibition term from the maximum Gabor Enrgy response: Equation (16).

$$b^{I,\alpha}_{\lambda,\sigma}(x,y) = H(\hat{E}_{\lambda,\sigma}(x,y) - \alpha t^I_{\lambda,\sigma}(x,Y)) \tag{16}$$

We fix α at 1 again.

2.2.1 Postprocessing

To obtain the final contour maps we need to make two standard postprocessing steps. Nonmaximum suppression and hysteresis thresholding[2].

The contour detectors which we described return images, in which the edge thickness is dependent on the size of the filter used. Nonmaximum suppression uses the orientation map obtained in the previous step to thin the edges to 1 pixel thickness. The method checks the two neighbours according to the orthogonal direction of the pixel orientation of each pixel. If either of the neighbours has a higher value than the middle pixel, the middle one will be set to 0.

Hysteresis thresholding is the final step in which the amount of false edges is reduced. Two thresholds are needed, a high threshold t_h and a low threshold t_l, for which $t_h > t_l$ holds. We fix the inequality at $t_h = 2t_l$. First we threshold the result of nonmaximum suppression by t_h and t_l to obtain Im_h and Im_l respectively. Then we we add all the nonzero pixels of Im_l which are neighbours of nonzero pixels in Im_h according to 8-connectedness.

The end result is a contourmap of the input image.

3 Spurious Edge Removal

We note that most edges, which are left by texture and still remain in the contour map, are not very long. These edges left by texture are the ones we need to eliminate. We propose two methods of spurious edge removal. One by checking the length of the connected components, and one by eroding the image with filters that contain straight lines.

3.1 Connected Components

Spurious edges generated by noise or texture typically consist of small groups. To reduce the number of spurious edges we look for small groups of connected components and then remove them from the image. Two edge pixels are considered to be in the same group of connected components if there is a path from pixel A to pixel B via other pixels. We look for connections in horizontal, vertical and diagonal directions. To determine if a group is small a threshold is specified.

Our algorithm first checks for every point that it is an edge points and not already tagged as a group, if it is connected to other edge points. The connected edge pixels are added to a stack to be checked later. All the connected points are given the same label. A more detailed description can be found in Algorithm 1

Algorithm 1 Connected Components Algorithm

```
for every edge pixel in the image do
  if the pixel is not labeled then
    put the pixel on the stack
    for every pixel in the stack do
      label the pixel with the label number
      for every surrounding edge pixel do
        if it is not on the stack and it is not labeled yet then
          put it on the stack
        end if
      end for
    end for
    increase the label number
  end if
  count the number of pixels per label
  remove the edge pixels of labels which have a size below a certain
  threshold
end for
```

3.2 Line Filters

To remove the spurious edges we erode the obtained contour map by a number of filters, each containing a straight line of a certain orientation. Then we reconstruct the image by dilation.
We define a linefilter as a square filter of odd size with a line through the origin which is the middle pixel. Let the originpixel be $(0,0)$ and let s be the size of the filter. For an orientation θ the line pixels are defined by (17).

$$l(i) = (i\,cos(\theta), i\,sin(\theta)) \tag{17}$$
$$\forall i\{i \in \{\lfloor -\sqrt{2}\frac{s}{2}\rfloor, ..., \lceil \sqrt{2}\frac{s}{2}\rceil\}|$$
$$\lceil \frac{-s}{2}\rceil \leq i\,cos(\theta) \leq \lfloor \frac{s}{2}\rfloor \wedge \lceil \frac{-s}{2}\rceil \leq i\,sin(\theta) \leq \lfloor \frac{s}{2}\rfloor\}$$

With Equation (17) we have the pixels on a line through the origin and with orientation θ. The condition guarantee the line is defined within the filter and goes from one end to the other.
The reconstruction is done by dilation with square structuring elements. The size of the structuring element s_{se} determines how big the gaps in the edges to be reconstructed can be. Since we only want the edges which were picked up by the linefilters we fix the size of the structuring element to $s_{se} = 3$. This way there can be no gaps in the edges and only the connected pieces will be retrieved.

4 Experiments

We applied the previous described contour detection algorithms on the natural picture in Figure 2.

Fig. 2: The natural image used for experiments. This image is 480 × 640 pixels in size.

For the NRFI contour detector we used a filtersize of 51 pixels, roughly 10% of the image size. We use the following orientations for the Gabor Filter: $\frac{i-1}{N_\theta \pi}$ for $N_\theta = 16$ and $i \in \{1, ..., N_\theta\}$.

The Canny edge detection uses a Gausian filter of 5 x 5. The pixel fraction in the Hysteresis thresholding is set to 0.15.

We apply the spurious edge removal for different values of edge sizes. For the linefilters this means different filtersizes and thus different line lengths. For the connected components this means the number of the pixels in the connected components.

5 Results

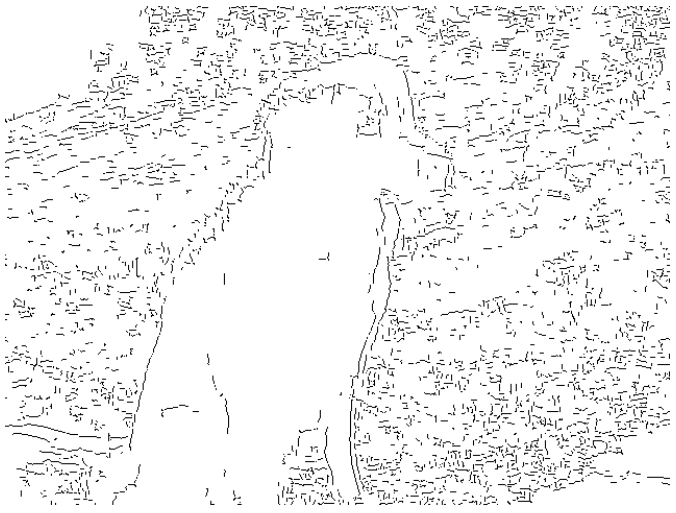
Here we discuss the results of the spurious edge removal, starting with connected components and ending with linefilters. The contour maps obtained by the contour detector using NRFI are shown in Figure 3, and we use these results for the spurious edge removal.

5.1 Connected Components

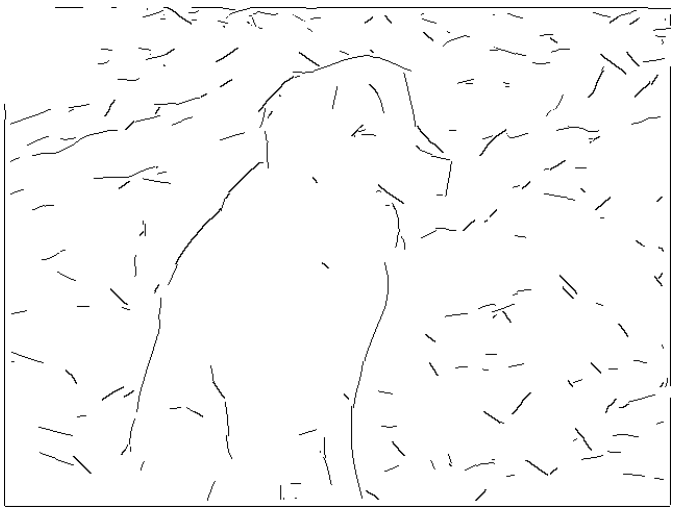
The results of using a filter based on connected components vary between edge detectors. Figure 4 shows the results of using the connected components filter after Canny Edge detection. Figure 5 and 6 show the results after isotropic and anisotropic NFRI respectively. When using the canny edge detector a low threshold does not remove a lot of the spurious edges, and increasing the threshold a lot of true edges are removed as well. When using the NRFI the results are a lot better. This is probably because NFRI removes a lot more spurious edges in advance then canny edge detection while retaining the true contours better. The threshold should be chosen carefully. Too low a threshold retains too many spurious edges, too high a threshold removes too many contour edges.

5.2 Linefilters

To make sure we would not miss any edges of a certain orientation we fix the number of orientations for the linefilters at $N_\theta = 64$. Again the results vary from contour detector to contour detector. In Figure 9 the contour maps of the Canny edge detector and linefilters show, many spurious edges still remain. Figures 7 and 8 show less edge noise but still either many spurious edges remain or many contour edges are removed. This is due to the nature of the contour edges. These edges are not always straight. Often they are curved. A large enough linefilter will not respond to curved contours. Connected components might be better suited for this task. A larger linefilter size decreases the spurious edges but might also remove contour edges. This is analog to increasing the connected component threshold, which yields the same results.



(a) Canny

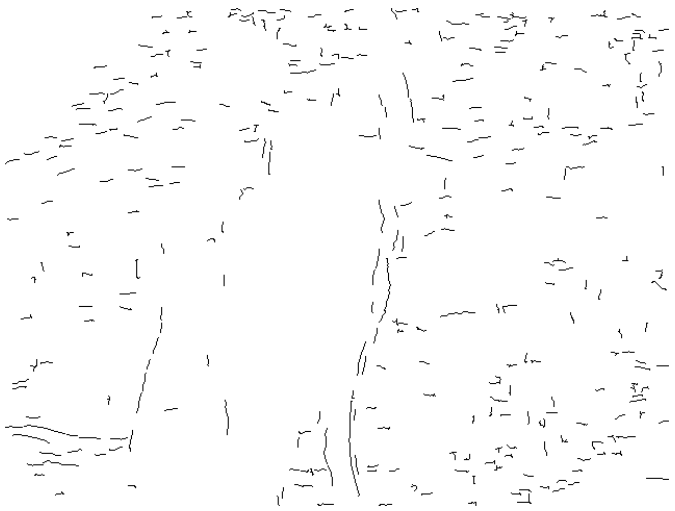


(b) isotropic

(c) anisotropic

Fig. 3: Contour maps obtained by applying edge detection.

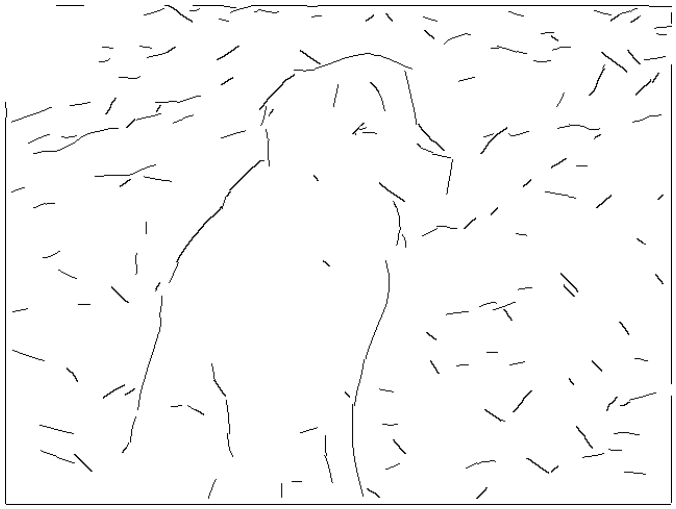
(a) Threshold = 5



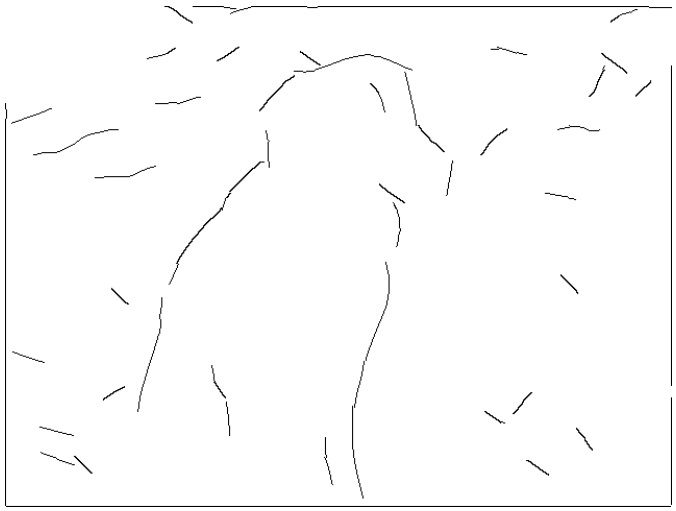
(b) Threshold = 10

(c) Threshold = 15

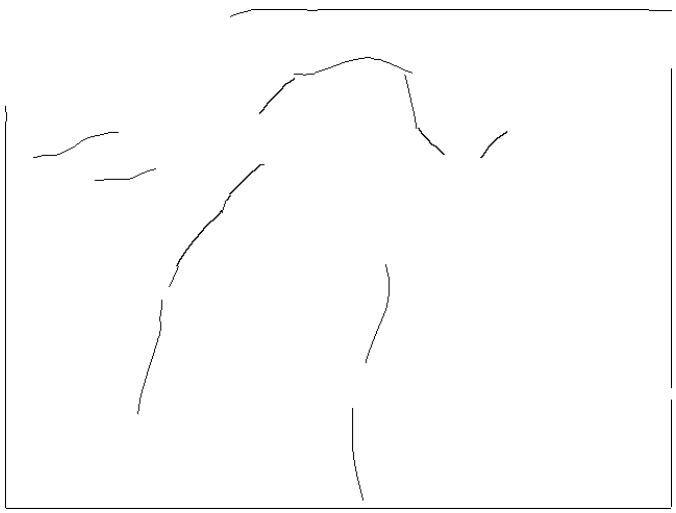
Fig. 4: Removal of connected components after Canny Edge Detection



(a) Threshold = 10



(b) Threshold = 30

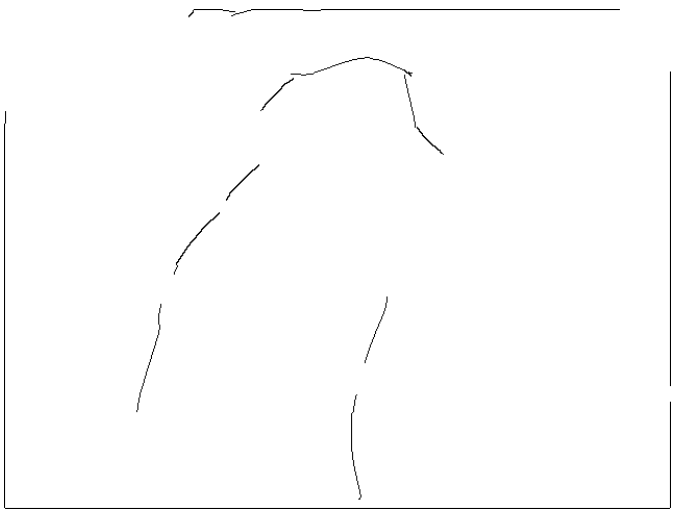


(c) Threshold = 50

Fig. 5: Contour maps obtained from isotropic NRFI and spurious edge removal by connected components. (5a):minimum connectedness: 10 pixels, (5b):minimum connectedness: 30 pixels, (5c):minimum connectedness: 50 pixels

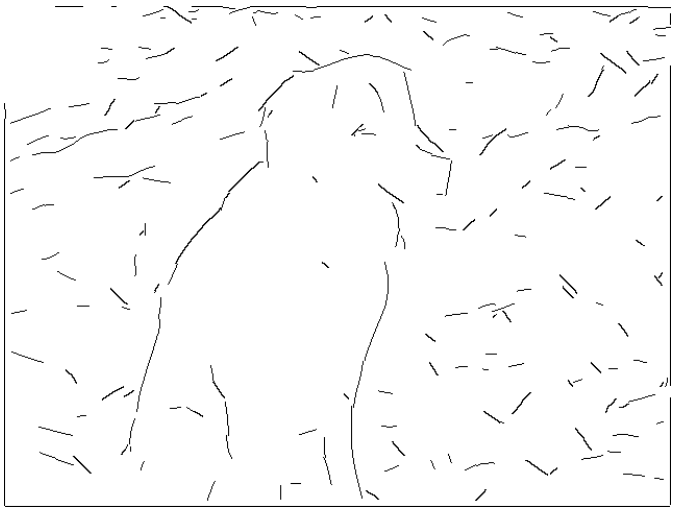
(a) Threshold = 10

(b) Threshold = 30



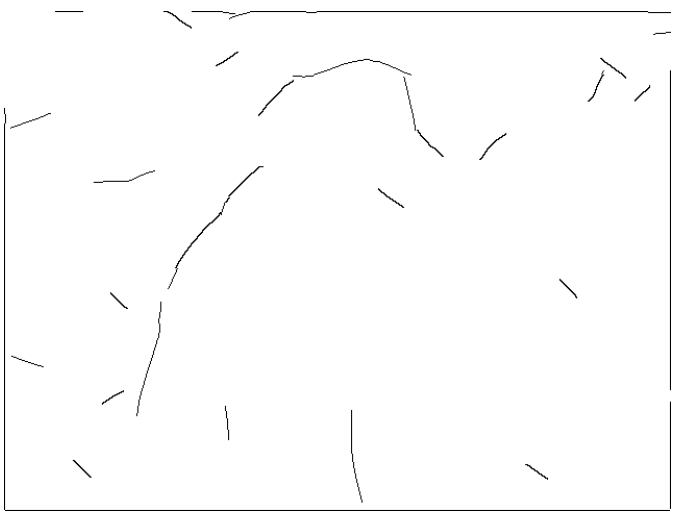
(c) Threshold = 50

Fig. 6: Contour maps obtained from anisotropic NRFI and spurious edge removal by connected components. (5a):minimum connectedness: 10 pixels, (5b):minimum connectedness: 30 pixels, (5c):minimum connectedness: 50 pixels



(a) Line filter size = 5

(b) Line filter size = 15

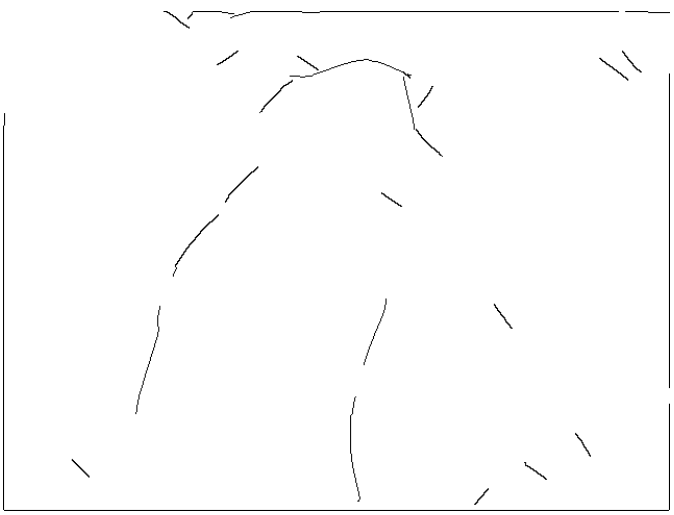


(c) Line filter size = 21

Fig. 7: Contour maps obtained by applying isotropic NRFI and spurious edge removal by linefiltering.

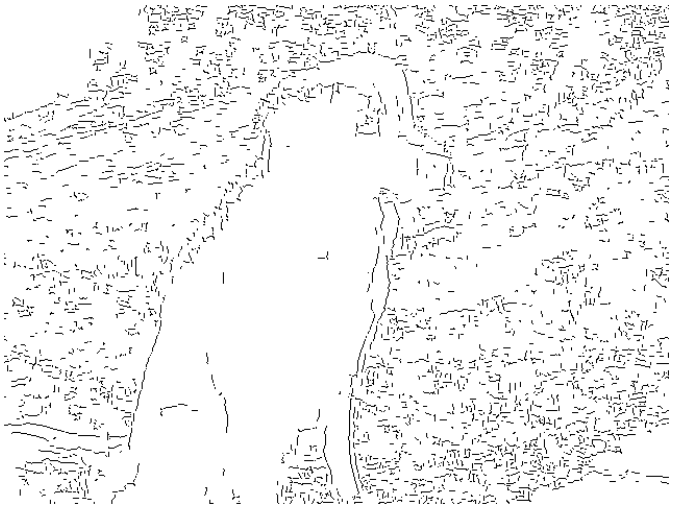
(a) Line filter size = 5

(b) Line filter size = 15

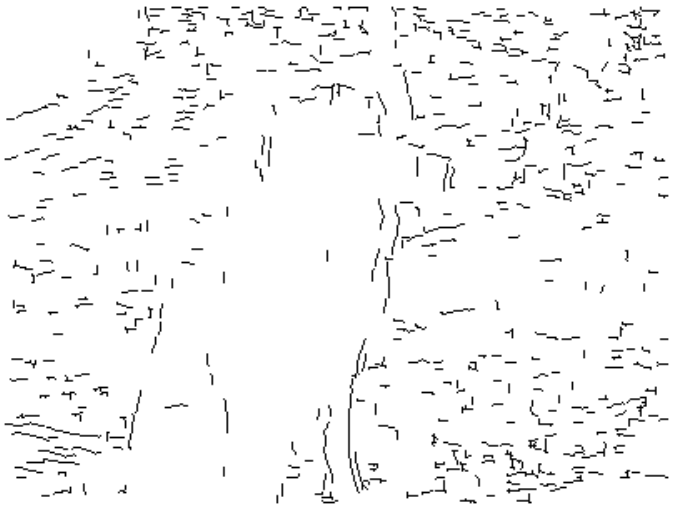


(c) Line filter size = 21

Fig. 8: Contour maps obtained by applying anisotropic NRFI and spurious edge removal by linefiltering.



(a) Canny no filter



(b) Line filter size = 5

(c) Line filter size = 11

Fig. 9: Contour maps obtained by applying Canny edge detection and spurious edge removal by linefiltering.

6 Conclusion and Discussion

Spurious edge removal through connected components yields better results than using linefilters. This is due to the fact that not all contours are straight, many are curved. Edge removal combined with the Canny edge detector works not as well as with NRFI.

The contours obtainded by contour detection using NRFI base filters tend to be longer than contours obtained by Canny edge detection and contain less noise of spurious edges. Since our initial observation was that most spurious edges are short, our approach is not suitable to combine with Canny edge detection. Choosing the edge size threshold is of great importance. Too small a threshold will have little effect. Too large a threshold will remove contour edges. The results we presented here show promise but more research is needed to reduce enough of the spurious edges without removing the contour edges. Possible subjects of further research are: using the combined edge strength of the connected components for thresholding; analyzing the edge lengths for automatic thresholding; the impact of the filter size on the edge response.

References

[1] E. Adelson and J. Bergen. Spatio-temporal energy models for the perception of motion. *J. Opt. Soc. Amer.A*, 2:248–299, 1985.

[2] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Pearson education, third edition, 2008.

[3] C. Grigorescu, N. Petkov, and M. A. Westenberg. Contour detection based on nonclassical receptive field inhibition. *IEEE Transactions on Image Processing*, 12(7):729–739, July 2003.

[4] J.G.Daugman. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *J. Opt. Soc. Amer.A*, 2(7):1160–1169, July 1985.

[5] M.C.Morone and D. Burr. Feature detection in human vision: a phase-dependent energy model. *Proc. R. Soc. Lond. B.*, 235:221–245, 1988.

[6] S.E.Grigorescu, N.Petkov, and P. Kruizinga. Comparison of texture features based on gabor filters. *IEEE Trans. Image Processing*, 11:1160–1167, Oct. 2002.

2D Touch Interfaces for Interaction with 3D environments

Frank Blaauw
University of Groningen
f.j.blaauw.1@student.rug.nl
s2051664

Wes Schuitema
University of Groningen
w.j.j.schuitema@student.rug.nl
s2075199

Abstract—For many scientific visualization problems such as particle simulations in astronomical data it is helpful to be able to view and explore a 3D environment/dataset; a lot of information can be derived from these visualizations. In addition to Virtual Reality approaches and other stereoscopic visualization settings such as *the CAVE* by Cruz et *al.* [4], people are also starting to explore the interaction using 2D touch-sensitive displays [8, 14, 16]. We analyze four different interaction methods and identify several concepts. A concept is an abstract description of a certain way of interaction; e.g., camera manipulation or object manipulation. Our research focuses on the similarities and differences of these concepts. The aim is to identify specific challenges and evaluate how these are addressed in the different interaction methods. The result of our research is a list of heuristics that can be applied to 3D interaction on 2D touch displays; i.e., which concept works well in certain situations and why.

Index Terms—Three-dimensional, Touch displays, interaction techniques, visualization, comparison of techniques

1 Introduction

We experience the world around us in three dimensions. Most of what we know, see, and use is three-dimensional. In contrast to this three-dimensional world, the computer many of us use every day does, in most cases, not provide these three-dimensions. Computer-based depiction is mostly two-dimensional.

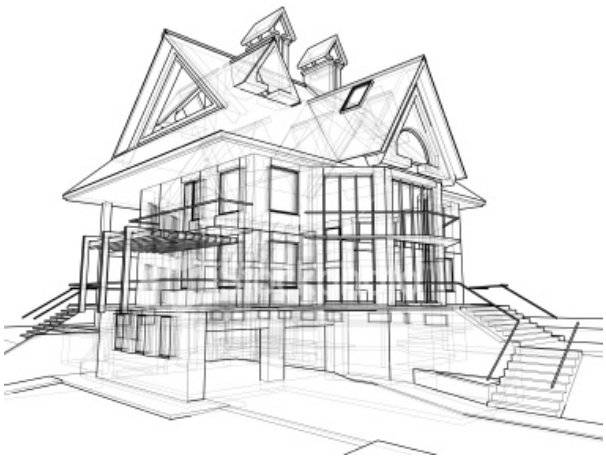


Figure 1. 3D technical drawing. From [13]

However, often it is still important to interact with a three-dimensional dataset, for example a technical design of a building as in Figure 1 where architects design a three-dimensional building, or in medicine, where datasets can provide insights in a patient's body. The problem that arises with these environments is that one needs to interact with three-dimensional objects using only two-dimensional input, such as a mouse or a touch screen. More specifically, moving up, down, left, and right is easily accomplished with two-dimensional input; it becomes more complicated when interaction with the third dimension is required, e.g., zooming or rotating an object in that third dimension. These actions *can* also be done in a two-dimensional environment, however for this the user needs more *degrees of freedom* (DOF). This means that some sort of mapping must be provided to achieve this interaction.

In this paper we focus on interaction using touch-sensitive input devices. More specifically, we identify a number of heuristics for three-dimensional interaction in a two-dimensional environment; i.e., what type of interaction works best in given situations.

To derive this list of heuristics, we first describe four different methods of interaction: The FI3D technique [16], the sticky tools technique [8], the simulation of grasping behavior [14, 15], and the four view technique in Section 2. By analyzing each of these different methods, we create a list of basic concepts; i.e., the basic techniques used to provide the interaction. In addition to this list of concepts we give an evaluation of each described concept. By evaluating the concepts, certain heuristics emerge; In Section 4 we present these heuristics together with their properties. This Section also provides a conclusion and a small summary of our findings.

2 Overview of Methods

There exist many approaches for interacting with 3D environments, such as special 3D mice or Virtual reality approaches, as shown by by Cruz et *al*. [4]. For this research we only focus on the use of multi-touch touch displays [5, 7] (or multi-touch screens) for the interaction. A touch screen provides, in contrast to, for example, a mouse, a way of interacting where a user can touch objects directly on a screen using his fingers or a pen rather than indirectly as with the mouse. However, there exist many techniques for doing interaction using a touch screen. In the following subsections, four of these techniques are described, each providing different approaches to provide the interaction.

2.1 Frame Interaction with 3D spaces

One technique for exploring 3D environments on a touch screen is the *FI3D technique* by Yu et *al*. [16] which is a direct-touch technique that allows users to explore three-dimensional data representations and visualization spaces. This technique focusses on cloud datasets or datasets with very many small objects in which there is not one dedicated object that can be used to constrain the mapping from to 2D input to 3D manipulation, such as an astronomical dataset shown in Figure 2. This approach uses a combination of *active regions* and a (multi-touch) gesture to provide the user with seven *degrees of freedom*. An active region in this case is a touch-sensitive region to which particular functionality can be attached. This means that the interac-

- *Frank Blaauw is a student of the University of Groningen, E-mail: f.j.blaauw.1@student.rug.nl.*
- *Wes Schuitema is a student of the University of Groningen, E-mail: w.j.j.schuitema@student.rug.nl.*

Manuscript received 31 March 2008; accepted 1 August 2008; posted online 19 October 2008; mailed on 13 October 2008.
For information on obtaining reprints of this article, please send e-mailto:tvcg@computer.org.

tion mode provided depends on the region in which the interaction is started. This type of interaction is called spring-loaded mode control [3]. The interface for the FI3D is shown in Figure 2.

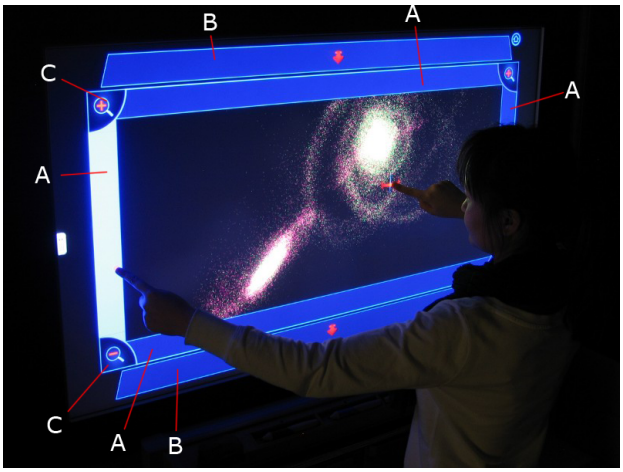


Figure 2. FI3D approach, by Yu et *al.* [16].

This technique provides x- and y-translation by dragging (i.e., the user can touch the dataset and move it over the x- and y-axis).

Rotation around the z-axis is done using the four active regions noted with *A* in Figure 2. Free rotation around the z-axis is provided by a virtual trackball. To use this virtual trackball, the user presses one of these four active regions and drags his or her finger over the screen. It is also possible to rotate the dataset around the z-dimension or only around the x- or y-axis. To do this the user moves his or her finger along one of the active regions for rotation around the z-axis or specifies the x- or y-axis using one of the regions.

To translate the dataset along the z-axis, i.e., bringing the dataset 'closer' to the user (not to be confused with zooming, where objects are enlarged on the same position on the z-axis), the user can use the regions noted as *B* in Figure 2. The user can select this region and drag a finger into the representation of the dataset. When dragging from the top area into the main representation, the user moves further into the dataset. Besides moving into the dataset, this same region also provides the user to move away from the dataset. For this the user does not release his or her finger and drags back up towards the top of the screen. This can also be done using the area at the button, however, this works the other way around. The dataset can be enlarged using the top areas marked by *C*. To scale down the image the opposite buttons are used.

These techniques provide the FI3D method with seven DOF. Besides these 'basic' interaction techniques, the technique also provides a gestural way for interaction. This gestural way combines four of the degrees of freedom, called *Rotate, Scale and Translate interaction* (*RST Interaction*) [9]. With this technique the user can rotate the dataset around the z- axis by touching and turning it with *two fingers*, but also translate the space in x- and y-direction as well as zoom into or out of the data. This is informally known as the *pinching gesture*, where the user moves his or her fingers closer to each other to enlarge the entire space, where the objects in the dataset are visually reduced, or further apart to reduce the size of the entire space, thus visually enlarging the image.

2.2 Sticky tools

The second technique we analyze is the *sticky tools* approach because it is designed to maintain the feeling of physical interaction with the full capabilities for the manipulation of a 3D scene [8]. To provide this feeling of physical interaction, three concepts are introduced: *sticky fingers*, *opposable thumbs*, and *virtual tools*.

Sticky fingers provides the user with four DOF. First of these is the ability to move an object around the x- and y-axis. This is done by dragging the object with one finger. This technique is illustrated in the left image in Figure 3. Besides moving, sticky fingers also provides the ability to rotate objects around the z-axis, by rotating it with two fingers. This technique is also shown in Figure 3. The last DOF provided is the ability to move an object along the z-axis, which is the *lifting* of objects. This movement along the z-axis and rotation is achieved using the *RST* technique (or pinching gesture) described in Section 2.1.

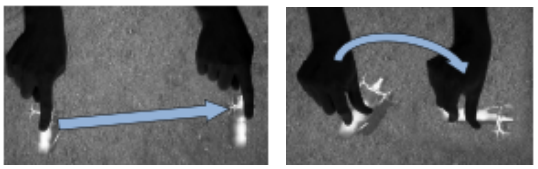


Figure 3. Movement and rotation using *sticky fingers*(in 2D), by Hancock et *al.* [8]

.

Because the sticky fingers concept does not provide enough degrees of freedom to maintain the feeling of physical interaction (for example, one does not have the ability to rotate objects), the opposable thumbs concept is introduced. This concept allows the user to rotate an object around the x- and y-axis by the use of a third finger, providing two additional degrees of freedom. For this technique the user places two fingers on the object and uses a thumb (or any other third finger) to drag away from the object, causing the object to rotate. An illustration of this technique is shown in Figure 4.

Figure 4. Sticky thumbs: rotation in 3D, by Hancock et *al.* [8]

.

The tools described earlier provide the technique with 6 DOFs. To add another DOF, sticky tools uses a technique called *virtual tools*. These tools are used to manipulate the continuous dimensions of the 3D objects, such as scaling the objects and applying textures to them. For example, the virtual tool to scale objects takes the form of a drawer. The user can drag an object into this drawer and can then adjust the scale of an object using a virtual knob.

2.3 Grasping

The third we discuss uses an interaction technique inspired by grasping [14] and is based on algorithms that can simulate grasping behavior on an imaging interactive touch surface. This technique uses a physics engine to simulate the behavior of the objects on the screen and lets the user interact with an object as if it is being grabbed. This means moving the objects happens in a similar way as you would do in a *real* 3D environment.

The technique provides this way of interaction by the use of proxy objects [15]. These objects are kinematically controlled to match the position of the surface contacts and can be thought of as virtual continuations of the physical contact points, i.e., where the fingers touch the surface. An illustration of this technique, with a representation of the proxy objects, can be found in Figure 5 (the proxy objects are the red circles in the image).

Moving the object happens in a way in which they would be moved in the real world. Objects on the surface can be moved along the x-and

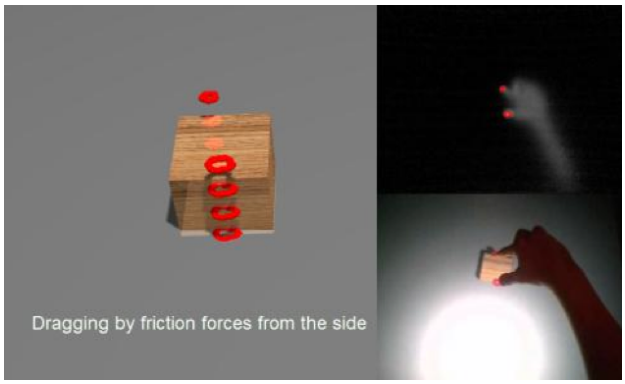


Figure 5. Grasping approach, by Wilson et *al.* [14]

y-axis by 'grabbing' them and moving them around in a way in which you would normally move real objects. The same applies to rotating objects, this can be done by, for example, grabbing it with two fingers and turning them.

This approach also has (some) support for a third dimension. The approach makes it possible to apply force to the objects. This means that a user can use a different amounts of force on the touch screen to *squeeze* or *press harder* on the object [14]. Combined with the physics engine, this can cause objects to be harder to move, as the friction between the object and the surface the object is on increases when force is applied to the object.

2.4 Four view

The *four view* method is one of the most commonly used approaches of interacting with 3D environments [10]. In this approach, the view on the subject is split up into four views; three views showing the 3D object in parallel projections along the three main coordinate axes, for example, from the top, the right side and the front, and a fourth view providing a perspective projection of the complete object. An example of this technique is shown in Figure 6.

Figure 6. Four view approach, from [12]

.

Each of the three views which show a 2D representation of the object enables the user to translate the object along two of the three axes, depending on which view the user is interacting with. For example, when the object is viewed from the top, it is possible to translate the image along the *x*- and *y*-axis. Moving along these axes can be done by touching and dragging the object. The fourth view is used to give an easy to understand 3D representation of the object.

3 Interaction Concepts

The goal of this research is to determine which type of interaction works well in which 3D environment. To evaluate the methods of interaction described in Section 2, we discuss this by the means of *interaction concepts*. These concepts provide an abstract description of the means to interact with 3D environments. This section describes and provides an evaluation of key characteristics of the concepts used in the four methods.

3.1 Camera vs. Object Manipulation

There are two different concepts for three dimensional interaction: camera and object manipulation. With object-manipulation, interaction takes place with the object itself. Camera-manipulation is a concept where the user can move the camera around. There is a big conceptual difference here: camera-manipulation lets users observe while object-manipulation lets users interact.

To provide a clear example of this difference in interaction, compare *FI3D* with *sticky tools*. The FI3D interface lets users manipulate the camera, e.g., users can zoom and pan the camera. The other method uses a fixed camera position, this leads to a different way of interacting.

With sticky tools there is not really a zoom function; if users want to view an object in greater detail, they can either make the object larger or lift the object so it is closer to the fixed camera.

Panning is used to view an object from a different perspective; there is a related way to achieve the same result using sticky tools. Instead of panning the camera to view another side of an object, the object itself can be rotated.

We evaluated both concept in order to identify important properties. The most important properties regarding camera-manipulation are:

- The interface only needs to supply the users with ways of changing the camera position. There is no need to have objects interact with each other or provide means to modify objects.
- This concept works well for applications where no interaction with an object is required; for example, walking trough a virtual house. The camera manipulation concept does not allow for interaction; for example, it is not possible to move around furniture in a virtual house.
- Camera manipulation only allows for one view at a time. This can be problematic when there are multiple users who each need to view something from a different perspective.

We have also identified some key characteristics of the object-manipulation, these are:

- Multiple users can interact at the same time, assuming the display supports multi-touch. Every user is able to manipulate a different object at the same time. In some situations, simultaneous interaction by multiple users and a single object is also possible; for example, two users can stretch or tear (Figure 7) an object when each user drags an edge of an object away from the other edge.
- Object-interaction is a very natural concept, it represents how people interact with real-world objects. There is, however, a drawback: there is no tactile feedback; i.e., you cannot feel the object.
- Object manipulation on small screens can cause problems. On smaller screens occlusion by the hands or fingers limit how precise people can interact; for example, when your finger is larger than a group of objects, it becomes difficult to interact with just one of these objects. The influence of occlusion on interaction is reduced when the screen becomes larger.
- In order to interact with an object, it needs to be near to the user. This can be problematic on large shared screens.

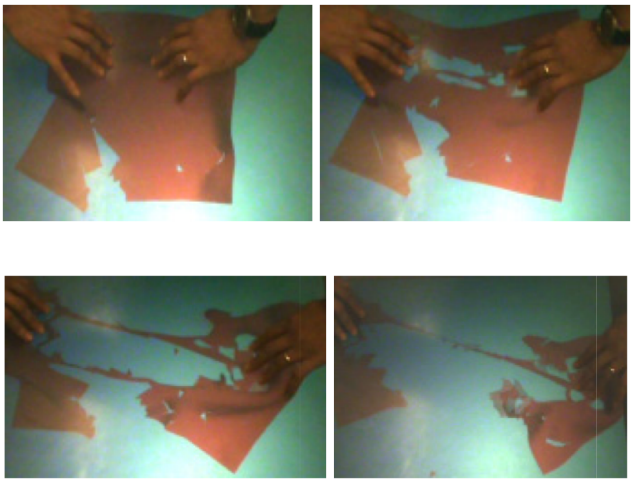


Figure 7. Multiple users interact on one object by tearing it, Wilson et *al.*[15]

3.2 Grabbing vs. Dragging

The displacement and manipulation of objects is an important form of interaction. A distinction can be made between two methods of moving objects; grabbing objects and dragging objects. Dragging is a commonly used method to move objects; this method is used in for example Apple's iPhone [6] and Google's Android [11] operating systems; these implementations, however, only provides two-dimensional interaction. Through the use of physics, extra degrees of freedom can be added to the dragging method. Another way to interact with objects is to simulate grabbing behavior. Grabbing is technically very similar to dragging; the difference is that instead of dragging an object, an object is pushed by invisible objects which, in turn, are dragged themselves. Although the methods are technically similar, the concepts are different.

The first concept provides the base for the sticky tools method. When touching an object with the sticky tools method, the user is able to drag an object. Grabbing behavior is used in the *Grasping* method. Objects here have a defined border that can be used to move an object; for example, a user can place his or her fingers around an object and subsequently move it. Both methods have a straightforward implementation when it comes to two-dimensional movement, three-dimensional movement is provided through the use of physics. The exact implementation of physics to achieve three-dimensional movement is discussed in the papers describing both methods [14, 15].

Both methods have been evaluated; the most important points are listed below, starting with grabbing. The most important characteristics of grabbing are:

- Grabbing an object is a very natural way to interact, this is what people do everyday.
- Interaction with a group of objects is possible; for example, a user can grab multiple objects at the same time.
- The use of physics can have unintended consequences; e.g., an object 'bumping' into another object, causing the object to topple.
- The lack of tactile feedback from the object is problematics; i.e. you don not feel the object, making interaction difficult.
- Grabbing only works in two dimensions, this means users cannot lift objects as is possible in real life.

The dragging concept is evaluated as well. We have identified the following characteristics:

- Using physics, users are also able to rotate and topple object, giving them more freedom.
- This is a natural way of interacting, e.g., users can slide object towards each other.
- Handling small objects can become difficult. There needs to be an area where the user can, for example, place his or her finger.
- Because a users needs to touch an area of the object, occlusion happens almost by design. Occlusion can cause problems; for example, some important part of the object cannot be seen.

3.3 Single- vs. Multi-View

Differences apply not only to the way interaction is achieved, there are also different ways to present information. When using a single view to display information it becomes more difficult to map which DOF to which changes in the system.

Another possibility is to use multiple views. These multiple views can be, for example, viewing an object from the side and from above. This multi-view approach has an advantage: the two-dimensional directions, up, down, left, and right then have a different meaning depending on the window, the interactions can stay two-dimensional. The multi-view approach is commonly used in computer aided drawing applications [1, 2]. It is common to also have a 'normal' three-dimensional view in conjunction with other views.

We have examined the characteristics of the multi-view approach, which are:

- This concept requires no extra mapping of the input to move along the x-, y-, and z-axis; this movement can be achieved with left, right, up, and down. This means that input from the screen can be directly mapped to the movement of the object.
- Some form of mapping of the input is still needed for scaling and rotation around the x-, y-, and z-axis.
- The impact of occlusion is reduced. What is occluded in a view can be clearly visible in another; for example, a view from above could let you see an object that is not visible in a side-view.
- The multi-view concept allows for very precise control, the movement of an object can be seen from multiple angles. This allows for precise placement, because depth can be observed in another view.
- Having to observe multiple views is not very natural. People are used to working from one viewpoint.
- Every view that is used needs to be visible on the screen. This means that more space is needed, or that each view has less space for itself, when using this concept.

3.4 Manipulating Continuous Dimensions

There are three ways for users to inspect an object in greater detail. The first option is to move an object closer to the camera (or the camera closer to the object). This is called z-translation. It is done regularly in real life; for example, holding a book closer to our eyes to read small text.

The second option is to make object *appear* larger, the object and camera stay in the same place. This is called zooming, this is also intuitive to most people. Zooming is also used in real life; for example, when zooming in on an object using a video camera or using a looking glass.

The third option may be a bit less intuitive, as it involves something that usually cannot be done in real life. This option is to actually make an object larger. When an object is made larger, its dimensions are manipulated; the addition of 'continuous' indicates that the new dimensions stay the same relative to the other objects. It should be noted that this is only relevant when working with multiple objects.

There are some things to keep in mind about this third option. It is a good option when there are multiple users, interacting with multiple objects, using a fixed camera position. When a user needs to inspect an object, he or she can make this object larger. However, this only works when there are other objects, in relation to which the object is made larger. When there is a single object, the results are, arguably, the same as zooming. This other object could simply be an indicator of the size of the object; for example, a ruler.

4 Conclusion

From the evaluation in Section 3, we can draw some conclusions. These conclusions enable us to create the following list of heuristics:

- **Just observation**: when the goal of the application is just observing an object or a set of objects, camera manipulation is the best choice. This is, for example, provided in the FI3D interaction technique. Here users can view the objects and move around them by changing the camera position. However, it is not possible to move or change an object itself. Also note that when using this technique with multiple users, each user sees the objects from the same viewpoint as the other users.

- **Interaction required**: interaction, in this context, means the manipulation of objects, e.g., moving them (separately) or scaling them. When the interface needs the users to interact with the object on the surface, the object manipulation concept provides the means to do this. This is in contrast to the camera-manipulation concept, in which the user only has the ability to view objects. It is possible, however, to combine the camera-manipulation and object-manipulation concept; this can be done, for example, by letting users switch with a spring-loaded mode [3]. This way the user can use camera-manipulation to view which object has to be manipulated, and switch to an object-manipulation technique to manipulate the object.

- **Multiple users**: for a screen providing functionality for multiple users, the best concept to use is object manipulation. This concept provides straightforward ways for multiple users to interact; users are able to, for example, move and mutate multiple objects. The camera manipulation technique could be used for multiple users as well, however, the users all see the objects from the same viewpoint. A solution for this could be splitting up the main screen in separately controlled smaller screens. Each user would then have his or her own sub-screen. Having the same viewpoint for multiple users is not necessarily a bad thing. However, in some cases it could be helpful, e.g., when someone is presenting information to the other people.

- **Small screen**: when a three-dimensional environment is displayed on a small screen (screen sizes smaller than 7 inches, e.g., mobile phones and tablets), the best observation concept is camera manipulation. The reason for this is that the object manipulation technique would be harder to use on small screens. This is because the objects become smaller and, therefore, become harder to touch and easier to occlude with fingers. This makes interaction with the objects harder. The impact of the occlusion in the camera manipulation is less because, after the interaction has been done, the fingers can be removed from the screen, so they do not occlude the objects on the screen. With this technique it is also possible to map actions to *active regions*, for example at the bottom of the screen. As you can use this region to control the interaction, you will not be occluding the main image. This is for example used in the FI3D technique [16].

- **View multiple objects**: a large dataset or multiple objects can best be viewed using the camera manipulation method. This method scales very well to the amount of objects in the set.

- **Screen orientation**: the correct concept to use also depends on the orientation of the screen. For example, when a screen using the object manipulation technique like sticky tools would be tilted, the technique would become less intuitive. Objects picked up would fall into the screen, instead of to the bottom and the mapping of the controls would be incorrect as well. Besides this, the viewpoint of the technique would be incorrect; it would be more intuitive to view objects from the side on a vertical display, instead of from the top.

  In this paper only object manipulation techniques for table top displays have been tested. It is however possible to use these techniques for vertical displays, however, the mapping of the controls, the view, and the physics should be adapted to the orientation.

- **High precision**: having multiple views is the best concept to choose when high precision is needed while interacting. This is mainly because this concept shows the object from multiple viewpoints, allowing the user to have a clear view on what happens when something is edited. This decreases the influence of occlusion between objects. Although the influence of occlusion by objects is decreased, occlusion by a finger becomes higher because the screen gets split into four sub-screens, meaning each windows is reduced in size significantly.

Acknowledgements

The authors wish to thank Dr. T. Isenberg, for giving his expert opinion and commenting on our paper. We would also wish to thank our peers K. Westra and S. Matyila for giving their opinion and reviewing our paper as well.

References

[1] Autodesk-inc. 3ds Max. http://autodesk.com/3ds-max/, Web site, visited March 2011.

[2] Autodesk-inc. AutoCAD. http://www.autodesk.com/autocad/, Web site, visited March 2011.

[3] W. A. S. Buxton. Human-computer interaction. chapter Chunking and phrasing and the design of human-computer dialogues, pages 494–499. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995.

[4] C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, R. V. Kenyon, and J. C. Hart. The CAVE: audio visual experience automatic virtual environment. *Commun. ACM*, 35:64–72, June 1992.

[5] P. Dietz and D. Leigh. DiamondTouch: a multi-user touch technology. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, UIST '01, pages 219–226, New York, NY, USA, 2001. ACM.

[6] J. Gonzalez-Sanchez and M. E. Chavez-Echeagaray. iPhone application development. In *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*, SPLASH '10, pages 321–322, New York, NY, USA, 2010. ACM.

[7] J. Y. Han. Low-cost multi-touch sensing through frustrated total internal reflection. In *Proceedings of the 18th annual ACM symposium on User interface software and technology*, UIST '05, pages 115–118, New York, NY, USA, 2005. ACM.

[8] M. Hancock, T. ten Cate, and S. Carpendale. Sticky tools: full 6DOF force-based interaction for multi-touch tables. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '09, pages 133–140, New York, NY, USA, 2009. ACM.

[9] M. S. Hancock, S. Carpendale, F. D. Vernier, D. Wigdor, and C. Shen. Rotation and Translation Mechanisms for Tabletop Interaction. In *Proceedings of the First IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, pages 79–88, Washington, DC, USA, 2006. IEEE Computer Society.

[10] A. Martinet, G. Casiez, and L. Grisoni. Design and Evaluation of 3D Positioning Techniques for Multi-touch Displays. Research Report RR-7015, INRIA, 2009.

[11] M. Sterk and M. A. C. Palacio. Virtual Globe on the Android – Remote vs. Local Rendering. In *Proceedings of the 2009 Sixth International Conference on Information Technology: New Generations*, pages 634–639, Washington, DC, USA, 2009. IEEE Computer Society.

[12] TurboSquid. Image used from the TurboSquid web site. http://www.turbosquid.com/FullPreview/Index.cfm/ID/581786, Web site, visited April 2011.

[13] Vladimir. iStockPhoto. http://www.istockphoto.com/stock-photo-5259104-house-3d-technical-draw.php, Web site, visited April 2011.

[14] A. D. Wilson. Simulating grasping behavior on an imaging interactive surface. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '09, pages 125–132, New York, NY, USA, 2009. ACM.

[15] A. D. Wilson, S. Izadi, O. Hilliges, A. Garcia-Mendoza, and D. Kirk. Bringing physics to the surface. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, UIST '08, pages 67–76, New York, NY, USA, 2008. ACM.

[16] L. Yu, P. Svetachov, P. Isenberg, M. H. Everts, and T. Isenberg. FI3D: Direct-Touch Interaction for the Exploration of 3D Scientific Visualization Spaces. *IEEE Transactions on Visualization and Computer Graphics*, 16:1613–1622, November 2010.

Multi-touch interaction on stacked objects in 3D environments

Jan-Paul Eikelenboom, Evert Kramer

Abstract—Creating an intuitive interface to manipulate 3D objects on a 2D display is an interesting but nontrivial problem. In this paper we explore and evaluate a number of different stack based approaches to interact with 3D objects. Different stacking strategies will be described along with gestures used in multi-touch applications and possible improvements to existing techniques. In our own approach we use a shallow-depth 3D environment, which is a 3D environment that can only be viewed from the top. The stack consists of several rectangle flat objects, which can represent images, documents, and other files. We will demonstrate that our implemented method provides improved ways to interact with a stack and, in our opinion, makes the user interaction easy, fast, and intuitive.

Index Terms—2D multi-touch , 3D visualization, Stacking, Intuitive interaction, Pilling strategies, Organizing objects

1 Introduction

Multi-touch and 3D environments is likely to play a greater role in future user interfaces. For instance multi-touch is implemented in the majority of hand-held devices and 3D techniques are implemented in more products than before. Tristram et al. [10] state that the future of user interfaces will use "3-D schemes that use our sense of spatial orientation to create the illusion of depth on-screen", instead of current desktop files, folders and icons. Thanks to these new techniques and approaches, people can interact much more efficient and intuitive with computers. Witthaker et al. [11] did research on the paper organization strategies 'piling' and 'filing'. The result was that piling is more efficient than filing. Using piling to organize objects has several benefits over filing, because it is easier prioritize, store, and find elements. The approaches that we explore try to solve this problem by using stacks on a multi-touch display and a shallow-depth 3D environment. Shallow-depth 3D [4] is a concept that is used in tabletop touch displays. This method allows interaction with 3D objects, but with limited depth. Hancock et al. [4] describes several interaction methods, which allow direct touch interaction on shallow-depth 3D objects. The results show that multiple-touch interaction offers more flexibility, speed, and accuracy. This concept makes it possible to simulate a more realistic stack interaction, by using multi-touch interaction, gestures, and a more natural 3D environment.

In this paper we first describe and evaluate the different stacking strategies. In the second part we explore the different approaches on how to interact with the stacked objects in a 3D environment. After evaluating and discussing the different methods, we show our own enhanced implementation of one of the methods and describe the improvements that we implemented, along with possible future improvements. At the end of the document we summarize the results we achieved with our implementation and compare it to existing methods.

2 Multi-touch interaction techniques

There already exist applications that offer multi-touch interaction and implemented different techniques to interact with stacked objects. In this section we discuss the approaches and interaction techniques of BumpTop [1] and Microsoft Surface[1].

- *Jan-Paul Eikelenboom is MSc. Computing Science student at the University of Groningen, e-mail: j.p.eikelenboom@student.rug.nl.*
- *Evert Kramer is MSc. Computing Science student at the University of Groningen, e-mail: e.kramer.1@student.rug.nl.*

[1]http://www.surface.com

2.1 Bumptop

BumpTop [1] is an application that offers different ways to interact with object stacks in a 3D desktop environment. In most multi-touch tabletop applications shallow-depth 3D is used, which allows interaction with limited depth and provides 2D interaction, but gives a 3D experience. The 2½D [1] viewing angle of BumpTop gives a better perspective of the workspace and gives the user a more natural environment to interact with. The lasso gesture is the main interaction technique with stack creation, in combination with a menu to select stack organizing actions. Instead of menus, it would be better to use different gestures for the most used functions, because of the extra effort it takes to select one in a menu. BumpTop examined what the best ways are to interact with stacks, by letting users evaluate the different techniques.

Examples of a contracted and expanded stack in BumpTop are shown in Figure 1. The stack is expanded by dragging the top object away from the stack. BumpTop uses a 2½D view on the desktop, which is a shifted perspective view of 25°, so that users can easily distinguish the size of the stacks.

Fig. 1: Example of a stack in BumpTop, image from [1].

2.1.1 Gestures

Interaction with stacks can be done in various ways, in this section we describe the most important multi-touch gestures. The gestures below are also visually shown in Figure 2. The animation between states of the objects and stacks is made as smooth as possible, with "Smooth Transitions" as the design goal. However, in some cases when an action needs to be more noticeable, the smooth transitions can be counter-productive. The gesture 'scrunch' is an intuitive and easy gesture to pile a selection of elements, that uses a smooth transition to pile the elements.

- Push: Objects can be added or removed from a stack, by moving an object on top of a stack or picking an object from an expanded stack and moving it out of the stack (not displayed).
- Fan-out: When two fingers swipe the top object of the stack outwards, the objects in the stack are fanned out following the object that is swiped away from the stack.

- Scrunch: When placing three or more fingers around a few scattered objects and then moving the fingers to the center of the objects, all those objects will be arranged in a stack.
- Lasso: Objects can be added to a stack by dragging a finger around a group of objects, when the lasso is closed the objects inside are arranged in a stack.
- Shove: With a larger contact area than just fingers, for example the side of a hand, objects can be pushed aside.
- Grow & Shrink: Objects and stacks can be zoomed in or out by using two fingers and moving them together or away from each other, also called pinch & zoom.
- Rotate View: When placing two fingers on an object, the object can be rotated by dragging the two fingers around each other.

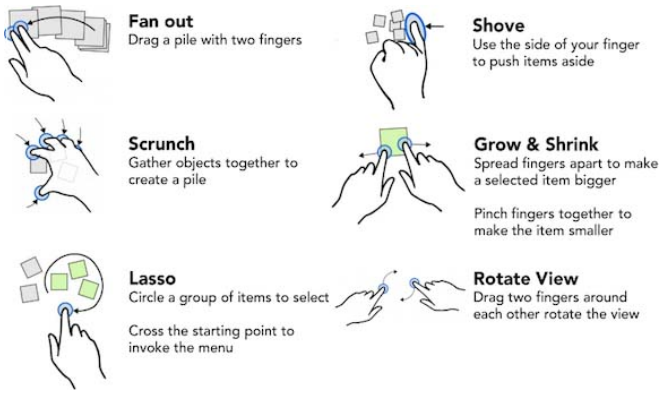


Fig. 2: Gestures used with stack interaction, image from [1].

2.2 Microsoft Surface

Microsoft Surface is an interactive multi-touch tabletop display, on which multiple users can interact simultaneously. The most important features, which set it apart from BumpTop are physical object recognition and multi user interaction. The Microsoft Surface tabletop display only offers a 2D environment, uses basic gestures and offers basic stack interaction techniques. BumpTop on the other hand offers more advanced interaction with 3D object stacks. Interaction with physical objects using Tangible User Interface (TUI) is an innovative aspect of this method, but only usable on tabletop displays. TUI makes the user experience more immersive, but requires intelligent stacking techniques to show the content of a tangible object and to allow the user to easily interact with the virtual objects.

Microsoft Surface works with stacks to order images, messages, and other objects. Stacks offer organization of contents and allow basic interaction. With the Microsoft Surface it is also possible to interface with a device placed on the surface, like a phone. After detecting the device, the Microsoft Surface can show the contents of the device in stacks. The stacks are visualized next to the device on the display, so that the user can interact with the content on the device. Offering more advanced interaction with stacks would give the user a more realistic experience and a faster way to interact with virtual objects [13]. Figure 3 shows direct-touch interaction on the Microsoft Surface tabletop display.

2.3 Tangible User Interface

TUI [9] is a new interface technique that links the virtual world with the physical world. The research into TUIs is still in an early phase, but is already used in some commercial products. Figure 4 shows how physical objects can function as handles, icons, and instruments to manipulate virtual objects, it also shows the mapping of physical TUI objects to virtual GUI elements.

Fig. 3: Interaction with Microsoft Surface using a TUI, image from http://www.surface.com.

Physical objects that can interact with the display are called tangible bits [6]. An example of this is content on physical devices like a phone or camera can interact with the Microsoft Surface. This way the user has more physical interaction with the display and can interact with the content on the device, this gives the user has a more immersive and realistic experience. Stacks can be grouped next to a tangible bit, for example figure 4 shows stacked photographs next to a device that the user can interact with.

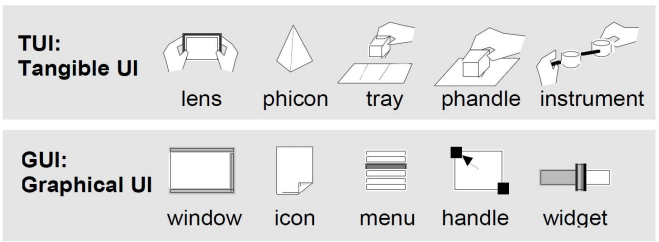


Fig. 4: Linking TUI to GUI elements, image from [6].

3 Other interaction techniques

Besides multi-touch, there are other techniques for interaction with virtual environments. The first is pen based interaction with stacks and the second uses multi-touch to simulate real world physics to interact with 3D objects.

3.1 Stacking in a 2D environment with a pen

This section describes a method for interaction on a 2D surface using a pen as input device. It is possible to fit the pen with sensors to provide hover and tilt [12] detection. Research in the field of simulating object stacks is already done using a digital table in combination with a pen [2]. This results in different strategies to interact with stacked objects. The principles behind these techniques can also be used to interact with stacks in a multi-touch environment.

Two of the strategies use transparency to reveal objects that are hidden in the stack. One of the advantages is that the remaining workspace is unaffected, when interacting with the stack. Transparency has the disadvantage that more calculations need to be performed when this technique is used. The third strategy allows interaction with an expanded stack (see Figure 5.c), the usable space on the environment is reduced, but the method is lightweight and more intuitive.

The first technique for stacking objects is called DragDeck, when the user wants to interact with the stack he or she just presses the pen onto the the top of the pile. Moving the pen across the surface causes the stack to become more transparent, revealing the hidden content. Selecting a revealed object and pressing the pen button moves the object onto the top of the stack. It is also possible to remove an item from the stack, by moving the pen orthogonally to the browsing direction.

This technique uses a hover interaction, instead of dragging to browse the stack, as shown in Figure 5.b. The image can be moved to the top of the stack by first selecting it and then removing the pen from the surface. Dragging an object results in removing it from the stack so that it can be used as a single object and new objects can be dragged onto an existing stack to be added on top.

In the second technique the stack is expanded. When the user touches the stack, the objects then are scaled to fit the workspace, see Figure 5. In expanded state, the user can reform the stack by clicking on an empty space, or on another object, the latter results in the selected object being placed on top of the selected stack. Removing an object is done by dragging it out of the expanded stack. Adding an object to the stack works the same way as in the other techniques.

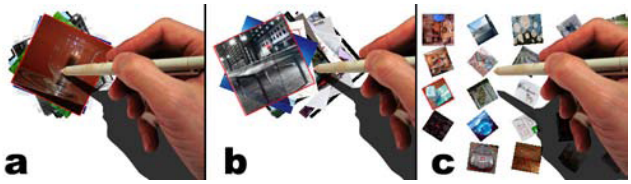


Fig. 5: Different states of a stack. In *a* the stack is in the normal state, in *b* the stack is in browsing mode and in *c* the stack is open as in the ExpandPile method, image from [2].

3.2 Analysis

Each of the techniques proposed for the 2D environment offer the same interactions, but with a different approach. In each technique a touch gesture is used to open the stack. The browse action requires a different action in each technique, as with repositioning a stack. Reorganizing the stack is done with either a drag or lift interaction, depending on the technique used.

To find out which one is the best technique out of the three, the authors of the paper included an usability study [2]. In the study, test subjects where asked to perform three different tasks that are supported by each technique:

- Navigate: Browsing trough a stack to find an image with a certain shape on it
- Repositioning: Finding images, showing one and two, then put them in order on top of the stack
- Reorganizing: Comparing stacks and find the same image, then move it from one stack to the other

It turned out that HoverDeck scored higher, than the other techniques, except in navigating through a stack (see Figure 6). Which is logical, because the ExpandPile reveals all elements by expanding, making it easy to find an element. However, most people preferred the HoverDeck techniques although it was on average slower than the other techniques.

3.3 Stacking with physics in a 3D environment

Physics play an important role in our daily life and dictate the manner in which objects interact with the environment. Which in turn gives users a certain expectation on how this interaction works. One of the challenges faced when trying to deliver an intuitive user experience, is to correctly simulate these interactions. Proxy objects [3] is a method which can simulate these interactions and could be used in stack interaction.

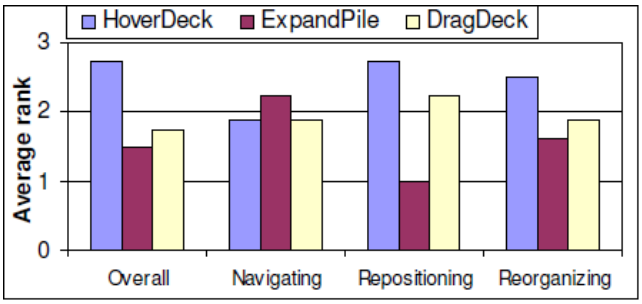


Fig. 6: Mean value for preferred approach, image from [2].

3.3.1 Proxy objects

In most implementations of a multi-touch environment, a touch on the surface is directly translated into an 2D position. If there is an object at that 2D position in the virtual environment, the user can interact with it. Proxy objects [3] introduce a different approach, translating the touch on the surface into an object in the 3D environment. This approach is useful, because in reality people use their hands or fingers to interact with an object. For example, when rotating an object there should be a collision force from the side pushing the object away (see Figure 7), if we want it to work as expected.

Fig. 7: Different strategies for rotating an object within a multi-touch environment, image from [3].

For each touch on the surface a proxy object is created in the 3D environment, represented as a primitive, like a cube or sphere (see Figure 8). The primitive is placed in the 3D environment according to the location of the touch on the display. When the user moves his or her finger, the corresponding action is also processed for the proxy object. With this method the user can push objects around or grab them by placing two fingers on opposite sides.

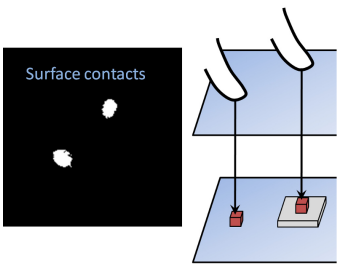


Fig. 8: Example of the Proxy Objects method, proxy objects are placed by ray casting, image from [3].

In the real world every object has a mass and can interact with other object by forces. This can be simulated using a physics engine, which also makes it possible to use proxy objects. With the help of a proxy object it is possible to simulate finger pressure onto an object, by

giving the corresponding proxy object a higher mass. This results in down force onto the virtual object beneath the proxy, in turn causing higher friction. Proxy objects make it possible to simulate friction and collision forces using an existing physics engine. This provides improved performance and reusable methods for rotating or grabbing objects, with the help of friction and collision forces.

The Proxy Object technique only handles single touch points, this introduces a set of limitations. These mainly concern the detection of complex shapes, like a hand or object placed onto the surface. Some interactive surfaces give enough information to detect the contour of the shape. With the use of Particle Proxies [3] it is possible to transfer the shape into the 3D environment. The basic idea behind Particle Proxies is to use a number of proxy objects to represent the shape, by removing or adding as much proxy objects as needed.

3.3.2 Stacking

The question is how can we use this technique to let the user create stacks. In reality, stacking a pile of papers is done by shoving it together with the hands until it forms a stack. However, in a 3D environment the same gesture would result in objects that represent documents [8] colliding into each other and flying everywhere, giving unrealistic results. The difference is that in reality a paper or document has some 3D shape, therefore making it possible to slide one on top of another. To make this possible in the 3D environment, Wilson et al. give the top and bottom surface of each object, a cambered shape. This shape makes it possible to tilt one object with a finger and sliding another object underneath, as shown in Figure 9.

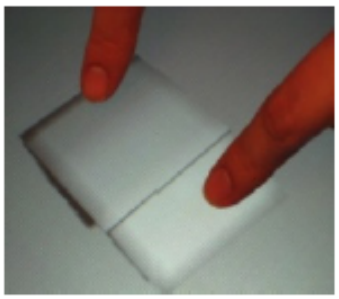


Fig. 9: Example of sliding one object underneath another, image from [3].

4 Analysis

In this section we compare the different techniques based on three criteria; input, interaction, and environment.

Technique	Input	Interaction	Environment
Bumptop	Multi-Touch	Gestures	Shallow-depth 3D with a 25 degree viewing angle
Microsoft Surface	Multi-Touch TUI	Gestures	Top view
Pen on tablet	Pen	Gestures and button control	Top view
Physics based	Multi-Touch	Physics	Shallow-depth 3D with top view

Table 1: Properties of the approaches for interaction with a digital environment

4.1 Input

In Table 1 we can see that the input for three of the techniques is similar, namely multi touch. Multi touch interaction offers users a natural way to interact with the surface by using multiple fingers, without the use of any tools. The pen on tablet technique also uses touch as input, but in the form of a pen instead of a finger. An advantage of using the pen, is that it can be fitted with different sensors, which add new functionality and interaction techniques to the touch interface. The downside is, that users need to know how to use the pen. How the different methods use gestures to improve the user interaction is discussed next.

4.2 Gestures

Our goal is to find an intuitive method for interaction with stacks. With intuitive interaction we mean that the user can interact with the stack in a similar way as with a real stack.

Bumptop uses different gestures to interact with stacks, these gestures have been extensively tested and analyzed, with respect to if they are intuitive and efficient. The main gestures that are used, lasso, fan out, and pinch & zoom are commonly accepted as ways to interact with multi-touch displays. Scrunch and shove are not as well-known, but sometimes are faster than other more simple gestures. These gestures are not as intuitive as lasso, for example, but they can make stack interaction much faster, so each method must decide on a trade-off between how intuitive and efficient a gesture should be.

Microsoft Surface is based on a 2D environment in which objects in the stack are assigned a different height, the gestures used are mostly based on one or two touches [13]. The stack interaction on the Microsoft Surface is not as advanced as the gestures and interaction used in BumpTop. The focus is more on simple object interaction and TUI, ”messy piles” [1] are used to stack objects. In a messy pile, objects are loosely stacked on top of each other in such a way that each object is partially visible. This only allows a few of the top objects on the stack to be visible, but messy stacks are suitable for the Microsoft Surface, which tries to keep interaction as simple as possible and the number of virtual objects to a minimum.

Pen based gestures can provide more accuracy, so it is easier to interact with elements in the stack than with the other touch-based gestures. Fanning out the stack and removing and adding items is more accurate, but it does not make the experience more realistic or intuitive. Using a pen is for most users not intuitive and working with fingers on a direct-touch display is more realistic and intuitive. When accuracy is needed in touch-based applications, pen based interaction is a good option, but for general touch-based applications for normal users, regular touch interaction is preferred.

Physics based interaction with objects is in reality very intuitive, but on a 2D display it is not intuitive for a user to interact the same way. There are not any gestures used in this method, but proxies can be used to provide a more intuitive way to slide objects out of an expanded stack. The proxies give a more precise control over the objects and this technique could be added to stack interaction gestures. However, a big disadvantage of physics simulation is that user could create unwanted collisions when interacting with multiple objects. The same holds with moving a stack because large stack could fall over when applying to much force, making it difficult to efficiently use the stack.

Another problem is that the system cannot detect the force a user applies, which in the real world results in less or more friction on an object. For example, sliding across an object with one finger will result in less force exercised than multiple fingers. It is possible to simulate this, by using Particle Objects to interact with the environment. However, it turned out that most users did not make this distinction, and just pressed hard on the surface to apply more force.

Our implementation uses the gestures fan-out and pinch & zoom to interact with the stack. These gestures make it possible to efficiently extract and add objects to the stack. Our method focuses on the extraction and addition of objects, while other methods focus more on visualizing the stack in different ways. We will not use the proxy method,

because in real world it is useful, but for working with stacks it is not efficient.

4.3 Environment

The last row of table 1 shows that either a 2D or 3D shallow-depth environment is used. Bumptop uses an 3D environment but has tilted the camera angle. The tilted camera makes it easier for users to see depth in the environment. Another trick Bumtop applies is the use of shadows to emphasize depth as described in earlier research [5]. Bumptop also offers walls were user can hang items on. The physics technique uses also a 3D environment but with a fixed top down view.

Microsoft Surface and the Pen on tablet technique both use a 2D environment. An advantage of using a 2D environment is that it is easy to translate finger-touches into coordinates for the 2D environment. Because the 2D environment lacks the dept dimension it is difficult to show a stack. To solve this, both techniques apply an drawing order, drawing the elements from to bottom to top. This will cause the top element to be drawn over the other elements, creating a depth illusion.

Both types of environments offer advantages and disadvantages. The 3D environment has a depth dimension, making it easy for the user to see depth and differences in size. Because the 2D environment only has the drawing order, to convey depth, it is harder to work with and to make the user aware of depth. However, the 3D environment may put more strain onto the system and the correct handling of touches on the surface is harder.

5 Stack Model for multi-touch environment

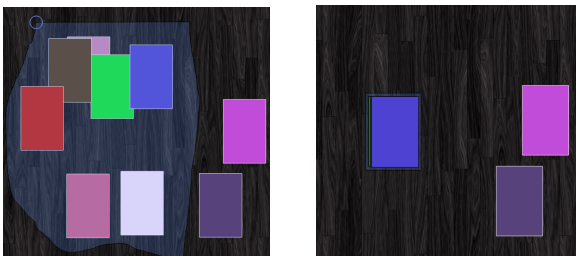
Based on the different stacking strategies, we can now propose a solution that takes advantage of multi-touch environments. There are a few basic functionalities which a stack needs in order to make it usable:

- Creating a stack from a group of objects
- Adding an item to the stack or adding a group of items to a stack
- Browsing the stack
- Reposition an object in the stack
- Removing an object from the stack
- Splitting a stack into multiple stacks.

Our implementation focuses on creating a stack, browsing the stack, and splitting a stack. We implemented our own method based on the discussed techniques to perform these tasks. In one technique physics are used, to interact with elements in the environment. Although this simulates real world interaction, there is a chance that unintended collision causes unwanted effects on interaction with stacks, hindering efficient use of the stacks in the environment. For the environment we use a 3D environment with a camera from above to make it similar to a desktop.

5.1 Creating a stack and adding objects

As explained in Section 2.1.1, a multi-touch environment recognizes gestures made by the user. In our model it is possible to create a stack with three different gestures. First a user can use the *Lasso* gesture, to make a lasso around the objects for the stack, as shown in Figure 10. An evaluation of people's multi-touch interaction behavior has previously shown that 56% [7] of the users used this method. As explained earlier, a *Scrunch* gesture can be used to gather objects by placing the fingers around the object and moving them closer to each other. When the fingers are released and the object distance between them is small, the group changes into a stack. A similar action is to use two hands to shove the objects together and when the objects are moved together close enough, the objects are turned into a stack.



(a) Step one (b) Step two

Fig. 10: Example of lasso action to create stack, Figure 10a shows first step, Figure 10b shows the result.

It is also possible to add more objects to a stack by dragging an object with a finger onto the stack. Objects are always added on top of the existing stack. The same holds for a group of objects that can be selected and dragged onto the stack, the objects are then added on top of the existing stack. With the *Lasso* gesture the user can make a circle around a group of objects, to select them and can then drag his finger to another stack to add those objects on top of the existing stack.

Our approach differs in two ways from the earlier discussed methods. Firstly, we automatically create a stack while Bumptop uses menus to specify what type of stack is created. Secondly, our stack elements are shifted, so that it is easier to see how high the stack is. Because we use a 3D environment with a top-down view it is difficult to see the height of a stack when objects are neatly stacked.

5.2 Browsing the stack and reposition objects

Browsing is an important functionality for a stack, because most of the time the user needs quick access to the stack. According to Aliakseyeu et al.'s work [2], HoverDeck provided the best experience. Keeping this in mind, we modeled our browsing method in a similar way by replacing the pen with a finger. In our model, the user can interact with the stack, by touching the top element of the stack and dragging it in any direction. While dragging the top element of the stack, hidden objects can be revealed (see Figure 11a), this is a technique similar to the *Fan out* gesture introduced in the BumpTop method.

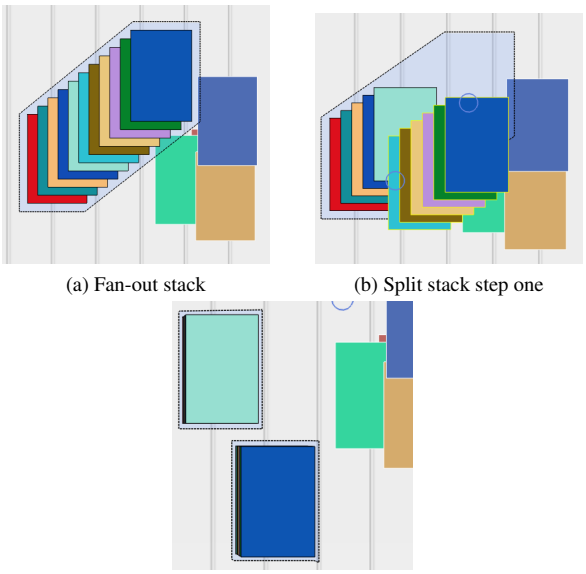
When an interesting object is found, another finger touch can be used to slide the object out of the stack, like a person would pick something out of a stack in reality. When the user removes his or her finger from the surface, the object will automatically slide back into the stack. Moving the finger while an object is selected further upwards or downwards on the stack results in repositioning the object in the stack.

5.3 Removing objects and splitting stacks

Removing objects works by touching an object and then dragging it out of the stack. The object is then released from the stack and can be moved free in the environment, or moved to another stack. Besides removing an object, it is also possible to split a stack using intuitive multi-touch possibilities. To split a stack the user needs to place two fingers onto two objects of an expanded stack. The objects selected and the objects in between are removed from the current stack when they are dragged away from the stack, these objects then form a new stack, as illustrated in Figure 11.

6 Conclusion

During our research we found different approaches in working with stacks, some of these approaches tried to model real world interaction, while other took a more abstract approach. It also turned out that physics do not always give a better result, when trying to achieve an user friendly method for interaction.



(a) Fan-out stack (b) Split stack step one

(c) Split stack step two

Fig. 11: Example of a split gesture.

We first thought that gestures should always to be intuitive, but a user evaluation [7] shows that a gesture, like the lasso, is more efficient and preferred by the users. So we can say that some gestures, in particular those that are widely used on interfaces are preferred over gestures that are more intuitive. Another important finding is that gestures can be divided in two groups, namely, command based or simulation based. The lasso is an example of a command based gesture and is a more abstract way to interact with the stack, while the scrunch gesture is simulation based and is more realistic and intuitive. In some cases it is best to use a combination of these groups, for example when it is widely accepted or efficient way to interact with a stack.

We can conclude that stacks are a useful method for interacting with objects on a desktop, our proposed method tries to make the interaction with a stack more intuitive and easy. We achieved this by modifying and adapting existing methods for interaction with a stack. We also introduced a new method for splitting a stack in a way that is similar to interaction with a real stack. Future work in the field of stack based interaction could be improving physics interaction, so that it adds value to the user interaction.

Acknowledgment

The authors wish to thank Dr. Tobias Isenberg, Drs. Jan-Mark S. Wams and Ing. Johan van der Geest for reviewing our paper.

References

[1] A. Agarawala and R. Balakrishnan. Keepin it real: Pushing the desktop metaphor with physics, piles and the pen. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, CHI '06, pages 1283–1292. ACM, Aug. 2009.

[2] D. Aliakseyeu, S. Subramanian, A. Lucero, and C. Gutwin. Interacting with piles of artifacts on digital tables. In *Proceedings of the working conference on Advanced visual interfaces*, AVI '06, pages 159–162. ACM, May 2006.

[3] W. Andrew D, S. Izadi, O. Hilliges, A. Garcia-Mendoza, and D. Kirk. Bringing physics to the surface. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, UIST '08, pages 67–76. ACM, Oct. 2008.

[4] M. Hancock, S. Carpendale, and A. Cockburn. Shallow-depth 3d interaction: Design and evaluation of one-, two- and three-touch techniques. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '07, pages 1147–1156. ACM, Apr. 2007.

[5] K. P. Herndon, R. C. Zeleznik, D. C. Robbins, D. B. Conner, S. S. Snibbe, and A. van Dam. Interactive shadows. In *Proceedings of the 5th annual ACM symposium on User interface software and technology*, UIST '92, pages 1–6. ACM, Nov. 1992.

[6] H. Ishii and B. Ullmer. Tangible bits: Towards seamless interfaces between people, bits and atoms. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '97, pages 234–241. ACM, Mar. 1997.

[7] C. North, T. Dwyer, B. Lee, D. Fisher, P. Isenberg, K. Inkpen, and G. Robertson. Understanding multi-touch manipulation for surface computing. In *Proceeding INTERACT '09 Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction: Part II*, INTERACT, pages 236–249. Springer-Verlag Berlin, Heidelberg, Aug. 2009.

[8] G. Robertson, M. Czerwinski, K. Larson, D. C. Robbins, D. Thiel, and M. van Dantzicht. Data mountain: using spatial memory for document management. In *Proceedings of the 11th annual ACM symposium on User interface software and Technology*, UIST '98, pages 153–162. ACM, Nov. 1998.

[9] O. Shaer and E. Hornecker. Tangible user interfaces: Past, present, and future directions. *Foundations and Trends in Human-Computer Interaction*, 3(1-2):1–137, Jan. 2010.

[10] C. Tristram. The next computer interface. page 1, Dec. 2001.

[11] S. Whittaker and J. Hirschberg. The character, value and management of personal paper archives. *ACM Transactions on Computer Human Interaction*, 8(2):150–170, June 2001.

[12] D. Wigdor and R. Balakrishnan. Tilttext: using tilt for text input to mobile phones. In *Proceedings of the 16th annual ACM symposium on User interface software and Technology*, UIST '03, pages 80–90. ACM, Nov. 2003.

[13] J. O. Wobbrock, M. R. Morris, and A. D. Wilson. User-defined gestures for surface computing. In *Proceedings of the 27th international conference on Human factors in computing systems*, CHI '09, pages 1083–1092. ACM, Apr. 2009.

COMPARISON OF SKELETON EXTRACTION TECHNIQUES

Marcel Jillings and Sijmon Heitmeijer
Department of Mathematics and Computer Science
Rijksuniversiteit Groningen, The Netherlands
m.jillings@student.rug.nl
s.heitmeijer@student.rug.nl

Abstract— In the field of skeleton extraction there are multiple algorithms which all have their own technique to extract a skeleton. This paper will compare three of these algorithms to help practitioners choose an algorithm which suits their needs and helps to determine global shortcomings of skeleton extraction techniques. The first algorithm is the Augmented Fast Marching Method which uses the observation that skeleton points are generated by the collapse of compact boundary segments during the Fast Marching Method algorithm's front evolution. The algorithm is augmented by a value U which is added to each point on the advancing front which allows the determination of the corresponding boundary point. The second algorithm is Computing Multiscale Curve and Surface Skeletons using a Global Importance Measure. Each point on the skeleton is assigned a part of the object surface, called the collapse. During the first phase of the algorithm the collapse measure is calculated, an importance measure where the size of the collapse is used for both curve and surface skeleton. Directional 3D Thinning using 8 Subiterations is the third algorithm being discussed. Thinning is an iterative process of layer by layer erosion of an object to extract an approximation to its skeleton. There are two types of 3D thinning algorithms: the curve-thinning type is used to extract medial lines or centerlines, whereas a surface-thinning type produces medial surfaces The inner workings of these algorithms are explained and the algorithms are compared by means of the robustness, complexity, ease of implementation and usability. The paper shows that the three different algorithms all produce usable skeletons but that robustness and ease of implementation are general issues. The Directional 3D thinning method can't handle noise and doesn't provide an implementation. The Global Importance Measure and Augmented Fast Marching Method give better results but the latter provides code segments, a ready to use implementation and is less complex.

Index Terms—skeleton extraction, centerlines, directional thinning, global importance measure, augmented fast marching method

1 Introduction

Extraction of skeletons is a fundamental problem with many applications in computer graphics and visualization. The skeleton of an 3D object consists, in general, of curves and surfaces [4]. A curve skeleton is a set of 1D curves that are locally symmetric with respect to the shape boundary [5] whereas the surface skeleton is a 2D set union of surfaces and curves [6].

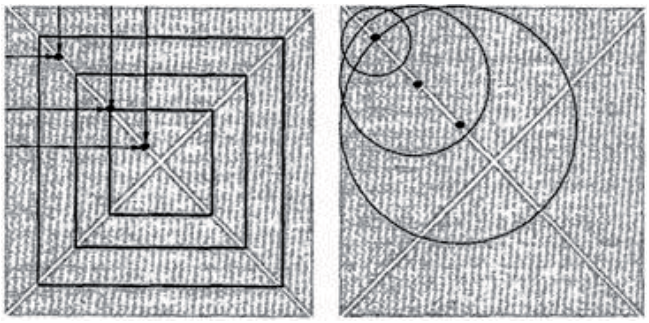
Curve skeletons have a low dimensionality which is useful for many visualization tasks such as virtual navigation, reduced-model formulation, visualization improvement, animation, geometric processing and morphing [2, 12]. Skeleton surfaces are useful in many application areas, such as volumetric animation, surface smoothing and topological analysis used in shape recognition, registration, simplification and feature tracking [2].

There are many algorithms in the literature describing extraction algorithms for different applications. However, it is unclear how they compare to each other. In this paper, we provide an overview of three skeleton extraction techniques: Augmented Fast Marching Method, Global importance Measure , Directional 3D Thinning, and compare them based on certain comparison criteria. The comparison criteria are based on the goals the authors set for the algorithms. These goals also often affect the advantages and disadvantages of the algorithm. It is important to compare algorithms because of the wide scope of available techniques. A comparisons helps practitioners choose an algorithm which suits their needs and helps bring forth problems with current techniques that might otherwise have been overlooked or ignored.

The structure of this paper is as follows. In Section 2 we give a summary about how the different algorithms work. The comparison criteria for the algorithms are defined in Section 3. The criteria from section 3 are than used to compare all three algorithms in section 4. In Section 5 a conclusion is presented and contains suggestions for possible future work.

2 Skeleton Extraction Techniques

There are three different algorithms that we will compare. This section describes how the different algorithms work and note what the key features, according to the respective papers, of the algorithms are. These key features can be easy usability, noise resistance or real-time skeleton extraction.



a) b)

Figure 1: The advancing front of the FMM algorithm which determines the position of the skeleton points (a) and the maximal disks centered on the skeleton points (b). [9]

2.1 Augmented Fast Marching Method

This subsection will discuss the Augmented Fast Marching Method (FMM) algorithm as described by Alexandru Telea and Jarke J. van Wijk [1]. The key to their method is the observation that skeleton points are generated by the collapse of compact boundary segments during the front evolution of the FMM algorithm [9]. Figure 1a shows the inwards movement of the advancing front of the FMM algorithm. Black dots indicate skeleton points. In theory corresponds each skeleton point to at least two boundary points. Those boundary points are the boundary points touching the maximal disk centered

on the skeleton point (Figure 1b). For every point in the advancing front the corresponding boundary point is determined.

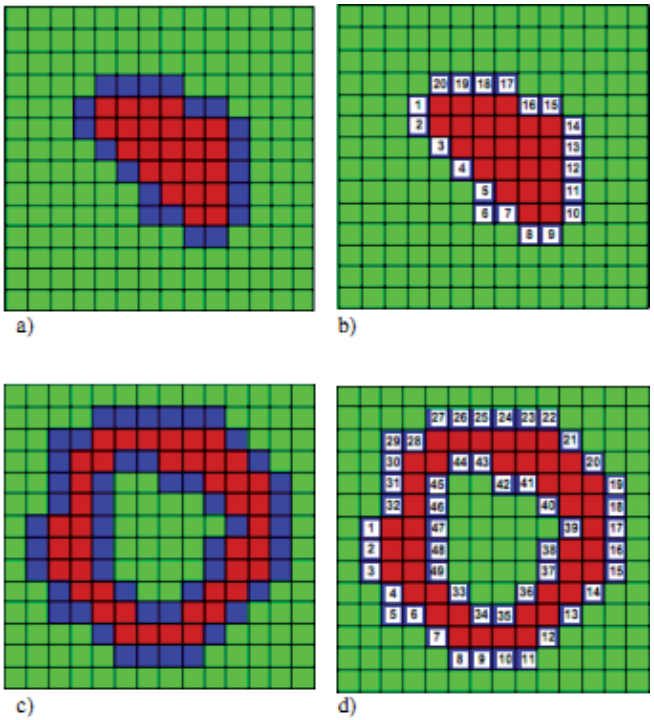


Figure 2: Objects (a ,c) and the order in which U is assigned to their boundaries (b, d). [1]

One real value *U* was augmented to the FMM algorithm per grid point in order to do this. Initially *U* is set to zero in an arbitrary boundary pixel. They assign a monotonically increasing *U* to every boundary pixel, equal to the distance, along the boundary, from that pixel to the *U = 0* pixel (Figure 2). This makes *U* a boundary parameterization with the property that the distance between any two boundary points, measured along the boundary, is equal to the *U* difference of that point. An exception to this are the points from which *U* starts being propagated, e.g. the point with *U = 1* in Figure 2b and the points *U = 1* and *U = 33* in Figure 2d. The *U* value is propagated along with the original FMM values. Every pixel inside the initial boundary gets marked by the propagation of *U* with the *U* value of the boundary point that arrived at that location on the advancing front. They interpolate *U* values, via averaging, to account for boundary points that are located between the initial boundary pixels. When the *U* values around the current point differ more than two it means that the boundary points were not neighbors. The maximum distance between two boundary points is √2 in case of diagonally connected pixels. The above happens along convex boundary segments that shrink (collapse) during front evolution in which case a skeleton point is found. A *U* field is computed for the

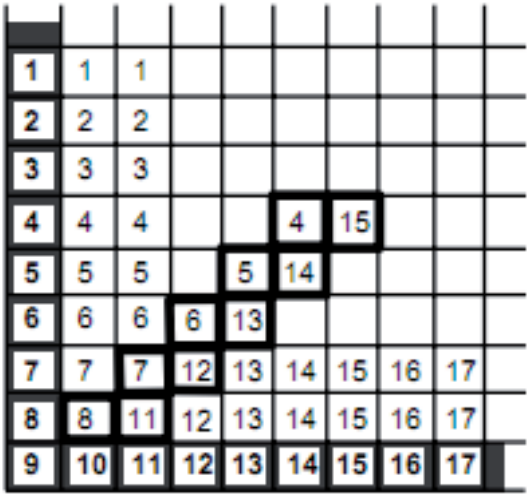


Figure 3: Detail of the augmented FMM result in the corner of a rectangle. [1]

figure on which a threshold is applied to retrieve the proper skeleton points. All points where *U* differs from the neighboring *U*'s by more than a given threshold remain.

Figure 3 shows the first three advancing fronts as evolved from the boundary of a rectangle. Note that the *U* difference between neighboring points increases as one goes further from the boundary.

Alexandru Telea and Jarke J. van Wijk explain that the property that *U* increases monotonically along the boundary's compact segments does not hold for the points in which they start the parameterization from. The problem of finding false skeleton branches at these starting points is solved by executing the whole augmented FMM algorithm twice and starting the *U* initialization from different boundary points. The intersection of the two resulting skeletons produces the correct skeletons in all the cases they tested.

2.2 Global Importance Measure

This section covers the Multiscale Curve and Surface Skeletons using a Global Importance Measure algorithm [2], which is designed to obtain both surface and/or curve skeleton hierarchies in a uniform manner. The algorithm is uniform because both, curve and surface skeletons, are computed and treated similar.

The algorithm consists of three separate phases. Each point on the skeleton is assigned a part of the object surface, called the collapse. During the first phase of the algorithm the collapse measure is calculated, an importance measure where the size of the collapse is used for both curve and surface skeleton.

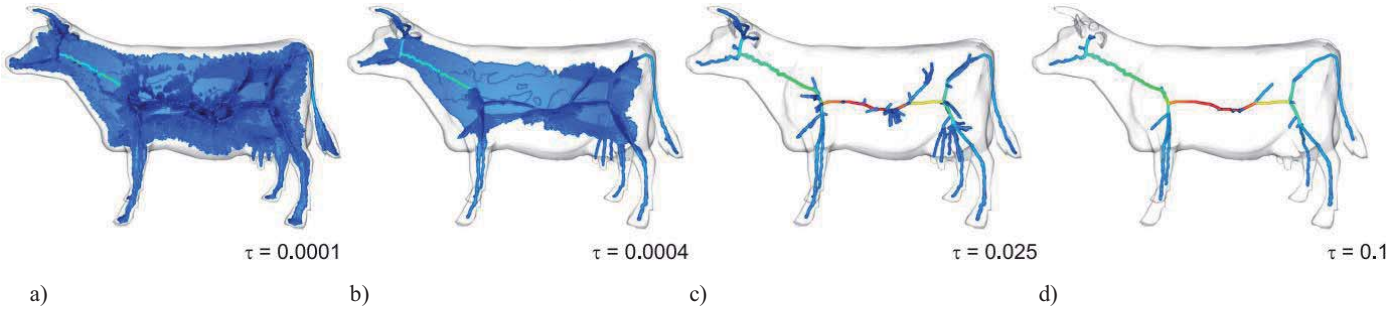


Figure 4: Example of skeleton extraction of a cow with multiple thresholds. [2]

Mass, initially located on the object boundary, is advected onto and then along the skeleton. The collapse measure of an object point is the amount of mass advected through that point. The amount of mass that passes through a point on its way to the root, the middle of skeleton, determines the importance of an object point. The collapse measure has a low value at the non-skeleton object points whereas the value increases while approaching the root.

Each skeleton point has at least two points on the shape's boundary at minimum distance, called feature points [7]. In the second phase the algorithm classifies an object point as a curve or surface skeleton point. Instead of only computing the shortest paths between both feature points of an object point, the feature points of the neighbors of the object point are also considered. This results in multiple shortest paths. The shortest paths form a band around the object. The object point is classified as a curve skeleton point if and only if the band splits the object surface into two connected components, assuming that the object is of genus 0. The object point is classified as a surface skeleton point if the band does not divide the object surface into multiple components.

Simplified skeletons can be obtained by pruning the skeleton using the importance measure [8]. After the collapse measure is calculated for each object point and the classification is done, the third phase is to obtain the simplified skeletons of the curve and surface skeleton. This is done by thresholding the collapse measures with a desired threshold. The collapse measures are normalized to *[0..1]* by dividing the values by its maximal value to easily handle objects of different maximum values. Because the collapse measure is monotonic the simplified skeletons are connected by default. The object points are of low importance and will disappear first when increasing the threshold, whereas the curve skeleton points are of high importance and will disappear last when increasing the threshold. An example of the algorithm using different thresholds is shown in Figure 4.

2.3 Directional 3D Thinning

This section will discuss the two 8-subiteration Directional 3D Thinning algorithms as described by Kálmán Palágya and Attila Kuba [3]. The goal of thinning is to reduce binary objects to their skeletons in a topology-preserving way. Thinning is an iterative process of a layer by layer erosion of an object to extract an approximation to its skeleton. There are two types of 3D thinning algorithms: the curve-thinning type is used to extract medial lines or centerlines, whereas a surface-thinning type produces medial surfaces.

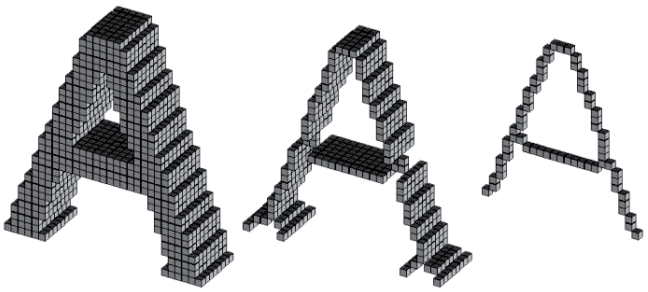
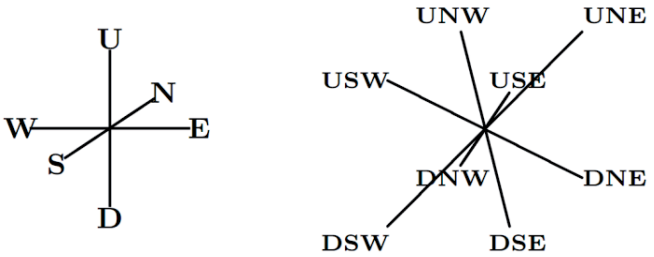


Figure 5: The original object (left), its medial surface (middle), and its medial lines (right). [11]

There are six kinds of border points in 3D images and 6-subiteration parallel thinning algorithms were generally proposed [9]. Instead of the six usual directions, eight new directions are proposed. These directions are depicted in Figure 6. The directions are labeled according to the four wind directions (N E S W) and as up and down (U D). Kálmán Palágya and Attila Kuba used some basic notions in their paper such as white points, black points, a black component, a white component, border points and simple points which we will not explain in detail in this paper.



a) b)

Figure 6: The usual six directions (a) and the proposed eight directions (b). [3]

The skeleton is extracted by deleting simple points from the image. A black point is called simple point if its deletion does not alter the topology [3]. At the start all points within the figure are black points (Figure 5, left) and the points outside the figure are white points. When a simple point is deleted it will become a white point.

In order to be able to retrieve a curve skeleton or a surface skeleton some black points are marked as curve-end points or surface-end points depending on the desired skeleton. These end points cannot be deleted. The following definitions are given:

Definition 1: A black point p is a curve-end point in a picture ($\mathbb{Z}^3$, 26, 6, B) if the set ($N_{26}(p) \setminus B$) $\cap$ {p} is singleton (i.e., p is 26-adjacent to exactly one black point).

Definition 2: A black point p is a surface-end point in a picture ($\mathbb{Z}^3$, 26, 6, B) if the set $N_6(p) \setminus B$) contains at least one opposite pair of points (Note that each curve{end point is a surface{end point.)

Seven base templates are assigned to the sub iterations of the curve thinning algorithm. These templates define the parallel reduction operation. A set of four base templates is assigned to the sub iterations of the surface thinning algorithm. These base templates are then rotated and reflected with respect to the three symmetry planes illustrated in Figure 7. Each direction gets its own template set, for curve thinning is this 22 templates per direction and for surface thinning this is 15 templates per direction.

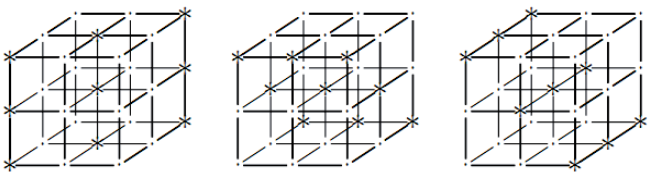


Figure 7: The three symmetry planes for reflecting templates. Points belonging to the given planes are marked "*". [3]

A black point can be deleted if at least one template in the set of templates matches it. By using different templates Kálmán Palágya and Attila Kuba are able to extract curve skeletons as well as surface skeletons.

3 Comparison Criteria

The paper presents a set of criteria which can be used for comparing the three algorithms from Section 2. When comparing different algorithms it is crucial to identify the criteria that lead themselves to comparison. The advantages and disadvantages of an algorithm are often influenced by the goals the authors set at the beginning of the research, and are usually described in the respective papers.

The different goals of the algorithms make it possible to determine the extent to which the goals have been attained or if they have been outperformed by other algorithms.

The Augmented Fast Marching Method algorithm has multiple goals. The main goal of the algorithm is to be easy to use and

implement for inexperienced users. The only parameter of the algorithm is a threshold value which is easy to choose by the users. The algorithm also sets a goal to behave robustly with respect to noisy boundaries. This goal is in line with the previous goal because other methods are very sensitive to their parameters' choice [1] with respect to the quality of the skeleton with noisy boundaries.

The main goal of the Global Importance Measure algorithm is to treat the non-skeleton, surface and curve skeleton points in a uniform manner. This should improve the overall running time because it uses a single global measure for both the curve and surface skeleton [2]. The secondary goal is that the curve skeleton is a subset of the surface skeleton. The curve skeleton can be considered as a limit case of the surface skeleton for objects with local circular symmetry [2].

The goal of the Directional 3D Thinning algorithm is to reduce binary objects to their skeletons in a topology-preserving way. A thinning algorithm does not preserve the topology if the object is split or deleted, any cavity is merged with the background or another cavity, or when a cavity is created which was not in the object.

All algorithms described above have their own goals. Some of these goals are in line with other goals and can also overlap with goals from other algorithms. Apart from the goals set by the authors a comparison is made for complexity, ease of implementation, robustness, usability and the quality of the results.

The complexity of an algorithm is the running time of the algorithm. There are problems having multiple algorithms with different complexity, while other problems having only one algorithm. The ease of implementation describes how much effort is required to implement the algorithm. Usability is the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use [10]. Robustness is the extent to which the algorithm is insensitive to boundary noise. This is required in all practical applications [2].

4 COMPARISON

We make a comparison between the different algorithms based on the comparison criteria from Section 3. A verdict is given at the end of each subsection.

4.1 Complexity

The discussion about the computational complexity will focus on the running time of the three methods.

The running time of the Augmented Fast Marching Method algorithm is for both 2D and 3D objects $O(n \log n)$ for n pixels (2D) or n voxels (3D).

The Global Importance Measure algorithm has a running time of $O(n(b \log b))$, where n is the number of object voxels and b the number of boundary voxels. The worst case scenario occurs when the object is a sphere, as the algorithm visits nearly all boundary voxels for diametrically-opposed object voxels [2], in which $b \approx \log n$. However, in most practical cases is the running time far below the worst case running time [2].

The last algorithm is the Directional 3D Thinning algorithm which has a running time of $O(n)$ for n voxels.

The computational complexity of the Directional 3D Thinning algorithm is the least of the three compared algorithms. The Augmented Fast Marching Method algorithm is the second least complex algorithm and the Global Importance Measure algorithm is the most complex algorithm.

4.2 Ease of implementation

As pointed out by Alexandru Telea and Jarke J. van Wijk, a lot of algorithms are very complex to understand and implement [1]. Most methods do not provide a detailed implementation or discussion on how to use their various parameters. The discussion about the ease of implementation will focus on the number of parameters, the implementation itself and documentation of the three algorithms.

The Augmented Fast Marching Method requires an image and a threshold as its parameters [1]. Code segments are given in the paper and a full working C++ implementation, including documentation, is available. These detailed code segments and a low number of parameters make the algorithm easy to implement.

The Global Importance Measure algorithm is comparable to the previous algorithm in terms of parameters. The algorithm requires an image and a threshold, to apply to the importance values, as parameters [2]. Pseudo code is provided in the paper but there is no actual implementation given. This makes it more difficult to implement than the previous algorithm.

The Directional 3D Thinning algorithm requires the least amount of parameters and only needs an image as the input [3]. Unlike the previous algorithms, there is a lack of any meaningful pseudo code. The paper also relies heavily on knowledge out of previous papers, which makes it difficult to understand the algorithm. Even though the algorithm only requires one parameter, the lack of code makes the algorithm by far the hardest to implement.

4.3 Robustness

In this section we will compare the robustness property of the different methods, based on the information offered in the corresponding papers. The Augmented Fast Marching Method algorithm and the Global Importance Measure algorithm both desire the robustness property because of the importance in practical use. The extent to which the algorithm is sensitive to boundary noise and be different for the surface and curve skeleton.

One of the goals of the Augmented Fast Marching Method algorithm is to be robust to noise. The collapse measure from the algorithm possesses this property. It is robust to compute on complex and noise

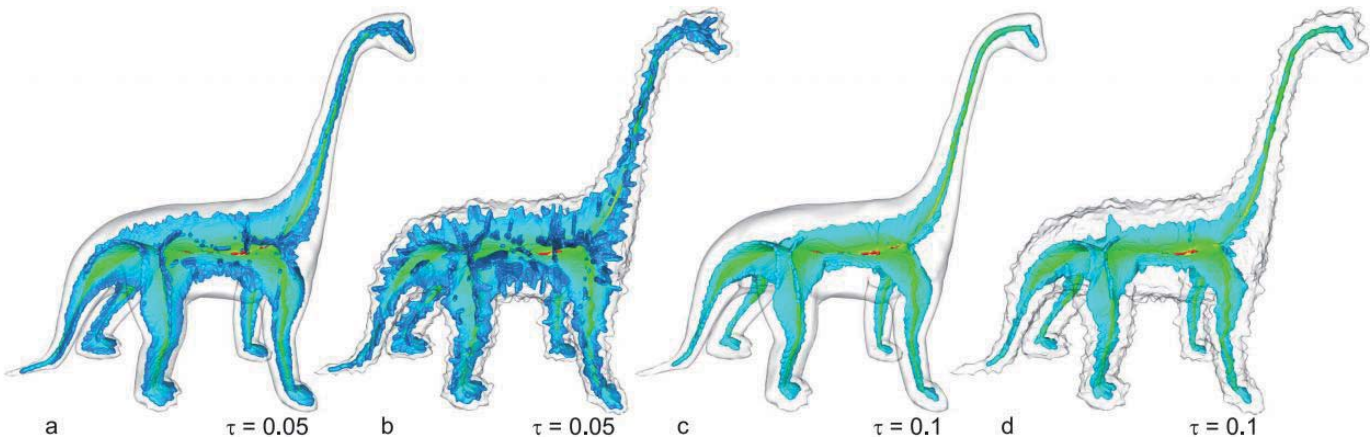


Figure 8: The skeletons can be made robust by increasing the threshold value. [1]

objects, and the result of a noisy surface is close to that of the corresponding smooth surface. This is achieved by setting the threshold value such that the skeleton parts due to noise are filtered out. In Figure 8 the dinosaur is depicted with and without surface noise with two different thresholds. The surface skeleton from the dinosaur with added noise (Figure 8b) is much noisier than the surface skeleton from the dinosaur without noise (Figure 8a).
Increasing the threshold value results in a more robust surface skeleton with noisy surfaces (Figure 8d). With the increased threshold the original dinosaur and the noisy dinosaur are comparable with each other. The curve skeleton and surface skeleton are calculated in a uniform manner, both the curve and surface skeleton use the same threshold value.

The Global Importance Measure algorithm presents a skeletonization algorithm which behaves robustly with respect to noisy boundaries. A pruning threshold t is used to prune skeleton branches, caused by boundary noise, shorter than t pixels. Good results with threshold values between 20 and 40 pixels on average are achieved for objects without added noise. For objects with boundary noise, the threshold values are increased to be around 100 pixels on average [2] to prune the small skeleton branches created.

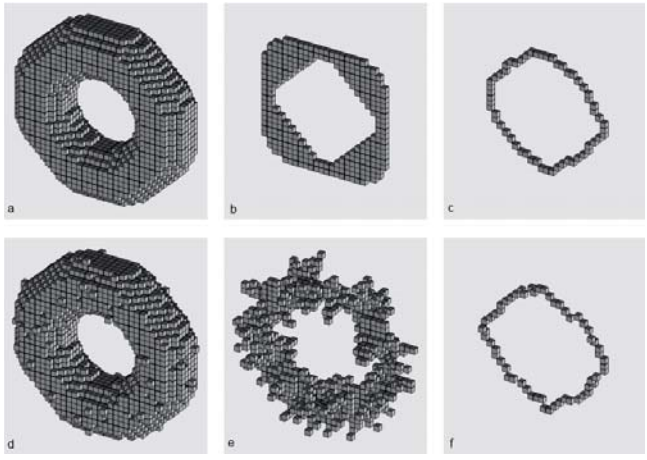


Figure 9: Noise sensitivity of the algorithm. The thinning of a solid doughnut (a) and its noisy version (b). The results of surface thinning (b, e) and the results of curve thinning (c, f). [3]

Unlike the other two algorithms, the Directional 3D Thinning algorithm only has the goal to reduce binary objects to their skeletons in a topology-preserving way. It is not a goal of the algorithm to be robust. In Figure 9 a solid doughnut is depicted in the original state and the noisy state. The noise in the noisy object was added to the boundary of the original object. Some border points are deleted from the original object and some white points adjacent to border points are changed to black. The surface skeleton from the doughnut with added noise (Figure 9e) is much noisier than the surface skeleton from the doughnut without noise (Figure 9b). This algorithm is sensitive to noise, since a noisy boundary may contain a number of surface-end points to be preserved [3]. Unlike the other two algorithms, it does not provide threshold values to reduce boundary noise. The curve skeleton from both the original doughnut (Figure 9c) and the noisy doughnut (Figure 9f) are comparable to each other.

The Augmented Fast Marching Method algorithm and the Global Importance Measure algorithm contain a threshold parameter to reduce the sensitivity to noise. The Directional 3D Thinning algorithm, which does not provide any threshold values to reduce the noise, is not suitable for surface skeletons but the curve skeleton of the noisy object is comparable to the curve skeleton of the original object. The results of the Augmented Fast Marching Method algorithm and the Global Importance Measure algorithm are similar in terms of sensitivity to noise, but the second algorithm is designed in particular for 2D objects. The Global Importance Measure algorithm can also be extended to 3D objects with additional implementation effort. The preferred algorithm depends on the skeleton type, the dimension of the object and the other desirable properties, like the complexity, usability and quality.

4.4 Usability

The usability of an algorithm depends on the objects being used. Noisy objects are widely used in practical situations whereas noise-free objects are rare [2]. In case of noisy objects the user want more control over the ability to reduce the noise.

The Augmented Fast Marching Method algorithm provides the user with a single threshold value to reduce object boundary noise, no other hacks or settings are needed and available. This threshold is intuitive and easy to use because the skeleton values are normalized to the *[0..1]* range by dividing the values by its maximum value. This algorithm is suitable for the use with noisy objects. There are no specific settings available for the skeleton extraction algorithm.

The Global Importance Measure algorithm is similar to the previous algorithm in terms of parameters. The algorithm provides the user with a single threshold value to reduce object boundary noise. Unlike the previous algorithm, the values are not normalized and more knowledge about the object is required in terms of size of the object [2]. There are no specific settings available for the skeleton extraction algorithm.

The Directional 3D Thinning algorithm has no parameters, hacks or settings available to reduce object boundary. Due to the lack of any settings this algorithm is sensitive to noise and the user will only be able to use it with noise-free objects for a surface skeleton. Curve skeletons are still useful for some objects but this depends heavily on the kind of shape which means that it can't be called robust.

None of the algorithms have settings to set the skeleton extraction algorithm to their preferences. This is not a problem for the users because the results of the algorithms correspond to the documentation. The Directional 3D Thinning algorithm has the worst usability due to lack of any settings.

The usability of the Directional 3D Thinning algorithm depends on the requirements of the user. If the user requires the possibility to simplify the skeletons with a threshold then it has the worst usability due to lack of a threshold setting. However, if the user does not require this setting, the algorithm has the best usability because the only requirement is the image. The first and second algorithm are identical in terms of settings, but the Augmented Fast Marching Method algorithm requires less information about the state of the object. The Augmented Fast Marching Method is thus the algorithm with the best usability when taking both settings and the amount of knowledge about the object are taken into account.

5 Conclusion & Future Work

This paper shows that the three different algorithms are all capable of producing usable and correct skeletons. The Augmented Fast Marching Method and Global Importance Measure algorithms are capable of handling noise unlike the 3D directional thinning algorithm.

Key features of the Augmented Fast Marching Method algorithm are the simplicity and usability of its algorithm, the provided implementation and the ability to adjust the threshold to fine-tune the resulting skeleton. It is also robust to noise and able to compute 3D centerlines but does this by intersecting the three 2D x, y and z planes.

The computation of skeletons using the Global Importance Measure algorithm has comparable results to the Augmented Fast Marching Method algorithm. It is robust to noise and thresholding the Global Importance value allows the user to influence the resulting skeleton. The algorithm is more complex then the Augmented Fast Marching Method algorithm but pseudo code is provided. This algorithm is capable of computing curve and surface

skeletons of 2D and 3D shapes. A limitation of the measure is that objects with tunnels cannot be handled.

The Directional 3D Thinning algorithm has the best complexity compared to the other algorithms however the paper in which it is presented relies heavily on previous knowledge and is hard to understand for non expert users. No implementation or pseudo code is provided. It is capable of computing 3D curve and surface skeletons but has no thresholding capabilities and fails in handling noise robustly.

Computing Multiscale Curve and Surface Skeletons with a Global Importance Measure seems to be the best choice because of its 3D capabilities and robust results. The Augmented Fast Marching Method algorithm is a good start for non expert users to gain experience or that have no need for full 3D support.

The main challenge is to include more, different and newer algorithms in the comparison. By including more algorithms it is interesting to see how different approaches and newer algorithms compare to each other using the defined comparison criteria.

It would also be interesting to implement the different algorithms and determine the difference between these algorithms for the defined comparison criteria over the same objects. Our current comparison between the algorithms is done only with the information provided in the respective papers.

References

[1] ALEXANDRU TELEA AND JARKE J. VAN WIJK, *An Augmented Fast Marching Method for Computing Skeletons and Centerlines*, Joint EUROGRAPHICS - IEEE TCVG Symposium on Visualization (2002).

[2] DENNIE RENIERS, JARKE J. VAN WIJK, MEMBER, IEEE, AND ALEXANDRU TELEA, *Computing Multiscale Curve and Surface Computing Multiscale Curve and Surface Global Importance Measure*, IEEE Transactions On Visualization And Computer Graphics.

[3] KÁLMÁN PALÁGYA AND ATTILA KUBA, *Directional 3D Thinning Using 8 Subiterations,* In Proceedings of the International Conference on Discrete Geometry for Computer Imagery, volume 1568 of Lecture Notes in Computer Science, pages 325-336, 1999. Springer Verlag.

[4] SEAN M. GELSTON AND DEBASISH DUTTA, Boundary surface recovery from skeleton curves and surfaces, November 1999.

[5] M. SABRY HASSOUNA AND ALY A. FARAG, On the Extraction of Curve Skeletons using Gradient Vector Flow.

[6] CARLO ARCELLI, GABRIELLA SANNITI DI BAJA AND LUCA SERINO, From 3D Discrete Surface Skeletons to Curve Skeletons.

[7] DENNIE RENIERS AND ALEXANDRU TELEA, Robust Segmentation of Voxel Shapes using Medial Surfaces.

[8] D. SHAKED AND A. BRUCKSTEIN, *Pruning medial axes,* Computer Vision and Image Understanding, vol. 69, no. 2, pp. 156–169, 1998.

[9] R. KIMMEL, D.SHAKED, N.KIRYATI, A.M BRUCKSTEIN, *Skeletonization via Distance Maps and Level Sets*, Computer Vision and Image Understanding, vol. 62, no. 3, pp. 382-391, 1995.

[10] ISO 9241-11, Ergonomic requirements for office work with visual display terminals (VDTs), pt 11, Guidance on usability, 1998.

[11] KÁLMÁN PALÁGYA AND ATTILA KUBA, *A 3D 6-subiteration thinning algorithm for extracting medial lines,* Pattern Recognition Letters, 19, pages 613-627, 1998.

[12] CORNEA, N.D.; SILVER, D.; MIN, P., Curve-Skeleton Properties, Applications, and Algorithms, 2007.

Performance Assessment of the Augmented Fast Marching Method for Two-Dimensional Skeletonization

Mark Scheeve and Karsten Westra

Abstract—The creation of a small shape representation of a 2D object is called skeletonization. Over the years several algorithms for computing skeletons were proposed. A. Telea and J.J. van Wijk claim that many of the known solutions use overly complex mathematical models and are difficult to implement and use. They proposed the Augmented Fast Marching Method (AFMM) as a more intuitive and simple approach to the challenge of skeletonization.
We assess the claims of the simplicity and ease-of-use by performing arbitrary experiments using the open-source implementation provided by Telea et al. The claim that the AFMM algorithm is simple to implement is assessed by investigating the translation from pseudo code to source code. The effectiveness is assessed by a comparison of skeletonization results using the AFMM algoritm and a thinning approach to skeletonization described by Gonzalez et al. [4]. Finally we discuss the results and present the conclusion of the assessment.

Index Terms—Skeletonization, 2D, Augmented Fast Marching Method, Visualization, Computer Graphics.

1 Introduction

A skeleton of a shape is a thin version of that shape that is equidistant to its boundaries. Skeletons provide a simple and compact representation while preserving the topological and size characteristics of the original shape. These characteristics make skeletons useful in many application areas e.g. medical visualization [12], computer vision [3, 10], object representation [6], path planning, computer animation and flow visualization [7]. Over the years many definitions of skeletons were proposed, but one of the first was given by Blum [1, 2] as the locus of the centers of maximal disks contained in the original object.
Skeletons can be produced in three main ways: Morphological thinning, geometric and distance transform (DT) methods.
Morphological thinning methods iteratively peel off the boundary layer by layer, identifying points whose removal does not affect the object's topology. These methods are relatively straight-forward, but depend on some complex heuristics to ensure the connectivity of the skeleton. These methods also do not produce a true skeleton by the definition given by Blum [1, 2].
Geometric methods make a polyline-like approximation of the shape boundary of the object and compute a Voronoi diagram. The resulting Voronoi diagram is the medial axis of the boundary [5, 6]. This produces an accurate connected skeleton, but are fairly complex to implement, are computationally expensive and require a robust boundary discretization.
The last class of methods computes the DT of the object's boundary.

- *Mark Scheeve is a student at the Rijksuniversiteit of Groningen, E-mail: m.scheeve@student.rug.nl.*
- *Karsten Westra is a student at the Rijksuniversiteit of Groningen, E-mail: k.westra@student.rug.nl.*

Paper written for Student Colloquium Conference given on 20 April 2011
For information on obtaining reprints of this article, please send e-mail to: k.westra@student.rug.nl or m.scheeve@student.rug.nl.

First of all we summarize related work in Section 2. In Section 3 we dive into the theory behind the AFMM method for finding skeletons proposed by Telea et al. Furthermore in Section 4 we provide implementation details of the algorithm with the help of pseudo code to help increasing the understanding of the algorithm. Then in Section 5 we use that understanding to aid in the comparison with a morphological thinning method and assess the claims of Telea et al. by performing a number of experiments. Finally in Sections 6 and 7 we have respectively our conclusions and a discussion including future work.

2 Related work

J. A. Sethian [8] introduced a robust and simple to implement algorithm for monotonically advancing fronts called the Fast Marching Method (FMM) which Telea et al. [11] used to create an algorithm for finding skeletons. A skeleton lies along the creases or curvature discontinuities in the DT. Finding these singularities in the DT is however difficult and many different solutions are proposed. Some methods try direct computation of the singularities, but this is a numerically unstable process and usually can not guarantee one-pixel-wide skeletons [7]. Direct singularity computation is attractive from a mathematical point-of-view but almost all methods fail to give a detailed implementation, performance analysis and discussion on how to set the parameters to achieve the desired results. Telea et al. present an algorithm which according to them is: simple to implement, well-performing, produces connected skeletons and is fast.

3 The Fast Marching Method

This section gives an extended summary of the theory that is used in [11] to help increase the understanding of the algorithm. The Augmented Fast Marching Method (AFMM) is a method for skeletonization proposed by A. Telea and J.J. van Wijk [11]. The AFMM is an augmentation to the Fast March-

ing Method (FMM) proposed by J.A. Sethian [8]. According to Telea et al. the rational behind using this FMM theory is that it is robust and easy to implement.

3.1 The FMM theory

The AFMM uses a mathematical theory based on the *distance transform (DT)* [1] method for computing a skeleton. The FMM algorithm computes a scalar field to solve the Eikonal equation

$$|\nabla T| = 1 \quad (1)$$

with $T = 0$ on the object's boundary. The field T is a good approximation of the distance to the boundary. The FMM algorithm uses an evolving front that moves from the boundary to the center of the image starting from the smallest T values. A so-called *narrow band* is maintained to keep track of this evolving front and is used to march inwards while freezing some of the computed T values in a particular case, hence the name Fast *Marching Method*.

3.1.1 Initialization of the grid points

We initialize the FMM algorithm by giving every 2D grid point a label and a T value. Depending on location the pixel and the evolving front it has three possible labels:

BAND: The point belongs to the evolving front, the so-called *narrow band* and the T value is undergoing update.

INSIDE: The point is inside the evolving front. Its T value is not yet known.

KNOWN: The point is behind the evolving front. Its T value is already known.

3.1.2 Propagation of the narrow band

After the initialization phase the FMM algorithm enters the propagation phase in which the T and f information gets propagated to the pixels inside the moving front. The order in which the algorithm examines these pixels is in ascending of the values for T. The propagation phase consists of four steps. Step one retrieves the coordinates (k,l) of the pixel with the smallest T value. Step two marches inwards. Step three computes $T(k,l)$ by solving Equation (1) in point (k,l) and step 4 finally adds the point with its recomputed value to the narrow band.
Step 3 solves Equation (1) by finite difference discretization on a Cartesian grid. The discretization of Equation (1) according to [8, 9] yields:

$$\max\left(D^{-x}T, -D^{+x}T, 0\right)^2 + \max\left(D^{-y}T, -D^{+y}T, 0\right)^2 = 1 \quad (2)$$

where $D^{-x}T(i,j) = T(i,j) - T(i-1,j))$ and $D^{+x}T(i,j) = T(i+1,j) - T(i,j))$ and similarly for the y direction. Telea et al. do not solve Equation (1) by solving Equation (2) for every pixel on the 2D grid, but they used a more efficient scheme as first introduced by Sethian in [8]. They solve Equation (2) for every 4 neighbors of the pixel (k,l) and use the solution which produces the lowest value for T.

[1] definition of distance transform: http://en.wikipedia.org/wiki/Distance_transform

3.2 Augmentation of the Fast Marching Method

Skeleton points are always generated by the 'collapsing' of compact boundary segments as the front advances [5]. The importance of a skeleton point is then given by the length of the boundary segment that collapsed into that point. The AFMM determines the origin of every point in the advancing front. Telea et al. use what they call a U value for all the points on the boundary. Initially all boundary pixels get a number starting with $U = 0$ in some arbitrary point and monotonically increasing U along the grid points as shown for objects a and c in Figure 1. The boundary will shrink when the fronts evolves. The new front will consists of fewer pixels that all get a new U value. This U value is assigned by setting the same value as that of their nearest neighbor in the current inwards moving front. In Figure 2 you can see an example of a collapsing front.

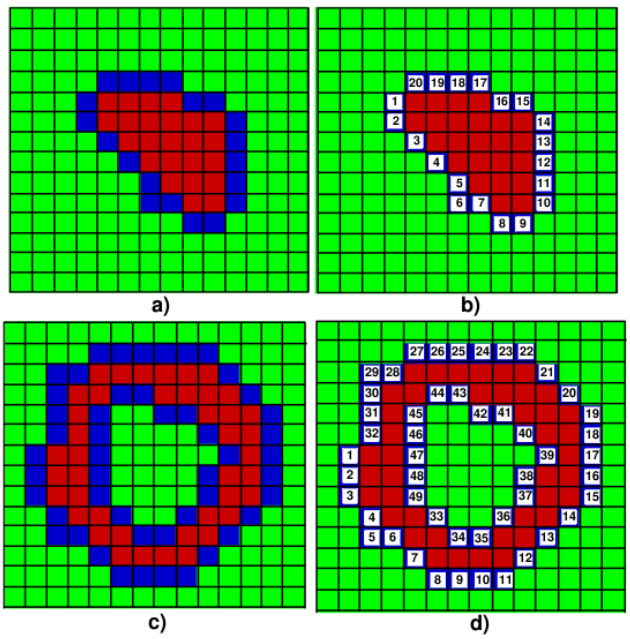


Fig. 1: *The order in which U is assigned to their boundaries. Taken from [11].*

The moving front initially consists of the the object's boundary with U values ranging from 1 to 17. The front will move inwards by iteratively solving the Eikonal equation for every pixel in the front starting with the lowest U value. Each pixel will be iteratively 'moved' to a pixel with the updated distance incrementation to the boundary. Moving in this case means assigning the U value from the previous boundary to the pixel that is currently under investigation. If a pixel already has a U value then the minimum of the current value and possible new value will become the new U value for this pixel. Now the only thing left to decide is when a new point is a skeleton point. As explained in Section 1 we want to find the singularities in the DT field. The U value differences provide us that ability. Knowing these locations of the singularities and their value differences allows us to use a threshold parameter t for pruning the skeleton branches which have a difference value lower than t.

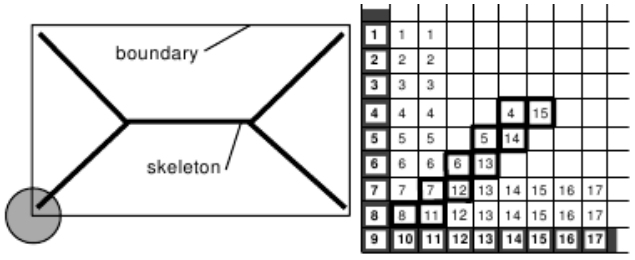


Fig. 2: *All the pixels on the boundary get a U value.*

4 THE ALGORITHM

The algorithm consists of two phases: the *initialization* and the *propagation* phase. In the initialization phase the DT map is created and every pixel gets a flag that points out if it is inside or outside the object or on the object's boundary as shown in Section 3.1.1. Each point iteratively is assigned a T and flag value f.

4.1 Initialization

When the current point is on the boundary its distance to the boundary is zero, it is assigned the BAND flag and the pixel is added to the narrow band. If the pixel is inside the object then its distance is set to MAX_VALUE and assign the $INSIDE$ flag. Telea et al. chose $10E6$ as an arbitrary maximum T value. Finally the distance of the points outside the object is set to zero and are assigned the $OUTSIDE$ flag. After the initialization phase the algorithm is ready for finding the singularities in the DT in the propagation phase. The initialization step, in pseudo code is described in listing 1.

Listing 1: *FMM initialization code as pointed out in [11]*

```
for all points (i,j)
  if ((i,j) on initial boundary)
  {
        f(i,j)=BAND; T(i,j)=0;
        add (i,j) to NarrowBand;
  }
  else if ((i,j) inside boundary)
  {
        f(i,j)=INSIDE; T(i,j)=MAX_VALUE;
  }
  else /* (i,j) outside boundary */
  {
        f(i,j)=KNOWN; T(i,j)=0;
  }
```

4.2 Propagation

After the initialization step the algorithm enters a propagation step. There are four steps to undertake in this phase. First the algorithm removes the first point P from the current front and gives the $KNOWN$ flag. Each point that is a neighbor of the point P is tested for the possibility of it being part of the new front. In step two the current possible "neighbor" under investigation gets the $BAND$ label if it is inside the object. In step three the algorithm starts to solve the Eikonal equation 1 and the obtained result is assigned to the T, or distance value, of point P. Finally the update version of the point P under investigation is inserted into the narrow band again. The propagation step in pseudo code can be found in listing 2.

Listing 2: *FMM iteration code as pointed out in [11]*

```
while (NarrowBand not empty)
{
  P(i,j)=head(NarrowBand);                    /* STEP 1 */
  remove P from NarrowBand;
  f(i,j)=KNOWN;
  for point(k,l) in {(i-1,j),(i,j-1),(i+1,j),(i,j+1)}
  if (f(k,l)!=KNOWN)
  {
    if (f(k,l)==INSIDE) f(k,l)=BAND;          /* STEP 2 */
    sol=MAX_VALUE;                            /* STEP 3 */
    solve(k-1,l,k,l-1,sol);
    solve(k+1,l,k,l-1,sol);
    solve(k-1,l,k,l+1,sol);
    solve(k+1,l,k,l+1,sol);
    T(k,l)=sol;
    insert (k,l) in NarrowBand;               /* STEP 4 */
  }
}

solve(int i1,int j1,int i2,int j2,float& sol)
{
  float r,s;
  if (f(i1,j1)==KNOWN)
    if (f(i2,j2)==KNOWN)
    {
      r = sqrt((2-(T(i1,j1)-T(i2,j2))
               * (T(i1,j1)-T(i2,j2)));
      s = (T(i1,j1)+T(i2,j2)-r)/2;
      if (s>=T(i1,j1) && s>=T(i2,j2)) sol=min(sol,s);
      else
      {
        s += r;
        if (s>=T(i1,j1) && s>=T(i2,j2))
          sol=min(sol,s);
      }
    }
    else sol=min(sol,1+T(i1,j1));
  else if (f(i2,j2)==KNOWN) sol=min(sol,1+T(i1,j2));
}
```

4.3 The solution

The solution skeleton we are looking for needs some more attention. We need to do thresholding and take care of an issue in the U values. These challenges are addressed in the following sections.

4.3.1 Thresholding

The difference in U value of each pixel will increase as the front evolves. When pixels get 'removed' the U difference to a neighbor of the 'removed' pixel will increase. When the U value difference becomes larger then that means that the pixels with their U values are not neighbors of each other. With this notion we can assume that pixels with large difference in their U value, which are not neighbors but that do collapse on each other, are interesting for our skeleton. An example of this is illustrated in figure 3. The pixel with $U = 4$ and $U = 15$, blue and red respectively, were indeed not neighbors in the original images. But after a couple of iterations in the AFMM algorithm they do collapse onto each other. The U difference gets larger when points are closer to the skeleton. A U distance threshold is chosen to obtain the best resulting skeleton. Telea et al. tested and validated the use of thresholding and came with a resulting threshold range of 20 to 40 pixels. Figure 4 shows the results from the validation by Telea et al.

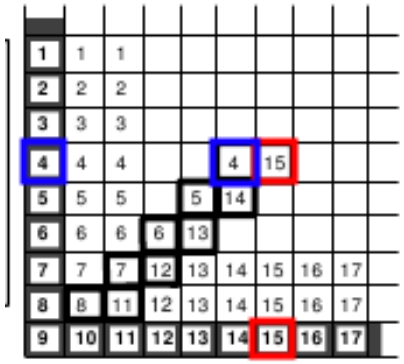


Fig. 3: *Threshold U difference. Taken from [11].*

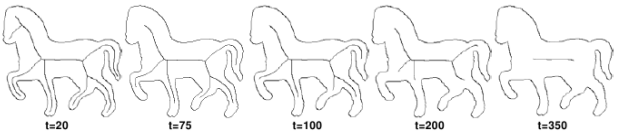


Fig. 4: *Threshold example. Taken from [11].*

4.3.2 Special case U value

Telea et al. mention that there is a 'special case' in which the U difference method does not result in the correct skeleton. The U values are assigned monotonically to all the pixes of the initial boundary. However the first and last pixel always have a large difference in their U value. This does not mean that they are always part of the skeleton that the algorithm is looking for. This 'special case' is illustrated in figure 5. This figure illustrates where boundaries are part of the skeleton where they really should not be part of it. Figure 5 illustrates were false skeleton edges might appear and shows:

a) the original object.

b) the initial U values.

c) the computed U values after AFMM run.

d) the resulting skeleton.

Telea et al. make the claim that it is easy to solve this special case. To overcome the problem they executed the AFMM method twice with a different starting point for assigning the U values. In this case they were able to remove the false edges by intersecting the skeletons from the two AFMM runs. All the tests of Telea et al. produced the correct skeletons that they were looking for.

5 Experiments

Some of the claims of Telea et al. in [11] are hard to quantify making them rather subjective. Subjective claims which are assessed and supported by many people are more likely to be accepted or rejected. We assess the claims of the simplicity and ease-of-use by performing arbitrary experiments using the open-source implementation provided by Telea et al. The claim that the AFMM algorithm is simple to implement is assessed by investigating the translation from pseudo code to source code. The effectiveness is assessed by a comparison of

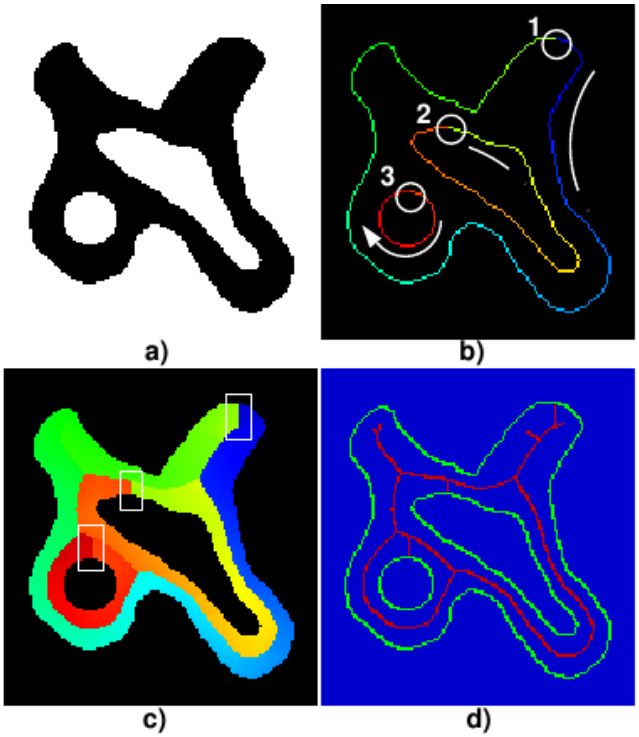


Fig. 5: *The special case. Taken from [11].*

skeletonization results using the AFMM algorithm and a thinning approach to skeletonization described by Gonzalez et al. in [4].

5.1 Simplicity and ease-of-use

The AFMM has one parameter. This parameter is a threshold which defines the minimum U value of the skeleton points as described in Section 4.3.1. The open-source AFMM implementation by Telea et al. allows to control the threshold directly. This makes finding the ideal threshold very intuitive, even for the non-expert users. To assess this, we have performed a number of tests to set the parameter for different images and the results are that according to us it works intuitive.

5.2 Implementation difficulty

Telea et al. provide an open-source implementation of their proposed algorithm. The AFMM is explained using pseudo code in section 4. We will now assess the claim of Telea et al. whether or not it really is easy to implement. The algorithm has been written in C++. Let us find out how much skill is required to implement/understand the solution of Telea et al.
The first step of the AFMM algorithm is the initialization step. In this step they need to assign a flag and a distance value to every point and add it to the *NARROW_BAND* if it is located on the boundary. This step is executed in the file *flags.cpp*. There is quite a lot going on in this class. The FLAGS constructor takes care of initialization. The current point under initialization gets a *ALIVE* flag, *KNOWN* in pseudo code. The distance value is set to $T = 0$. After this everything that is outside the boundary, and the object, gets a *FAR_AWAY* flag, *INSIDE* in the pseudo code. Its T value is set to *INFINITY*. After iterating over all points the remaining points which are *FAR_AWAY* and have a neighbor that are *ALIVE* then they add it to the narrow

band. It is a little bit different from the pseudo code. But the idea is fairly simple.
The second step is the propagation step. The theory behind this step is explained in section 4. This step is executed in the *mfmm.cpp* and *fmm.cpp*. The difference with the pseudo code and the code implementation is that the code implementation allows to use four different methods which are variants of the algorithm described in the pseudo code. The *fmm.cpp* file is 'responsible' for the four steps in the pseudo code. The augmentation, or modification, is implemented in *mfmm.cpp*. The five steps, four in pseudo code, are easily found in the `diffuse` method in *fmm.cpp*. The implementation gets the points with the smallest T values and its neighbors. If the current point is inside the shape then it is added to the narrow band. After this the eikonal equation 1 is solved by the algorithm.
When comparing the pseudo code to the actual source code, the first thing that is noticed is that the code is not a one-to-one translation of the pseudo code. This is probably because over the years many improvements have been made. The pseudo code seems relatively simple to implement, but because the open-source implementation was not a one-to-one translation and written in the C++ language, it was not that easy to read. However, we do believe that the implementation of the AFMM is relatively simple due to the pseudo code giving a much better understanding of the algorithm as a whole. Any decent programmer with the proper experience can make the translation to from pseudo code to source code.

5.3 Effectiveness

To assess the effectiveness claim we do a comparison of the performance in terms of speed and quality of the resulting skeleton. According to Telea et al. morphological filtering methods do not produce results which are in accordance with the definition given by Blum [1, 2] where a skeleton is the locus of the centers of maximal disks contained in the original object. For the AFMM we use the open-source implementation provided by Telea et al. and for the thinning method we use the Binary Thinning Image Filter (BTIF) which is implemented in the Insight Toolkit [2] framework. The BTIF is a sequential thinning algorithm and known to be computational time dependable on the image size. The algorithm corresponds with the 2D implementation described by Gonzales et al. in [4].

In Figure 6 we show the results of skeletonization between these two methods where the red line is produced by the BTIF and red line by the AFMM. Both skeletons are a good representation of the object from a human perspective. From the definition given by Blum the red line is the better skeleton of the two.
Another important property for a skeletonization method is the computational cost or speed. We test the two methods on the five different skeletonization problems on the same machine.

[2]The Insight Toolkit is available on http://itk.org/

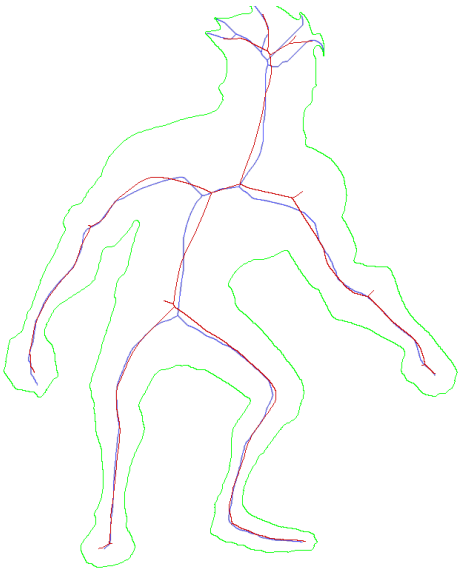


Fig. 6: *Skeletonization of cartoon.bmp. The red line is produced by the thinning algorithm and the blue by the AFMM.*

Table 1: *Comparison between AFMM and BTIF, tested on same machine.*

Image	Running time AFMM	Running time BTIF
anim5.bmp	0.249 sec.	3.510 sec.
camel.bmp	0.514 sec.	10.608 sec.
cartoon.bmp	2.852 sec.	57.938 sec.
leaf1.bmp	0.592 sec.	14.149 sec.
xxx.bmp	1.388 sec.	79.576 sec.

The outcome of this comparison shown in Table 1 shows that the AFMM computes a lot faster in comparison with the BTIF. Note that this particular implementation of a thinning method is not the most efficient thinning algorithm out there today.

Table 2: *Skeleton images used in comparison and corresponding number of pixels.*

Image	Number of pixels
anim5.bmp	76916
camel.bmp	156660
cartoon.bmp	842710
leaf1.bmp	182040
xxx.bmp	599536

Despite `xxx.bmp` having less pixels in comparison with `cartoon.bmp` the BTIF takes longer to compute a skeleton. This is due to the fact that the image `xxx.bmp` contains a very complex 'spatter-like' shape. The AFMM seems to be less sensitive to complex shapes.

6 Conclusion

With the introduction of the AFMM for skeletonization Telea et al. express their claims on the advantages of their method, but these claims are subjective. The thinning approaches have the advantage that they usually are fast to compute and easy to implement. The disadvantage is that these methods do not always produce a skeleton as the definition given by Blum [2] which may not be desirable. We used a thinning method implementation which was not really fast compared to the AFMM but is old and probably better solutions exist. Although we did not use Geometric methods in our comparison these methods in general produce good results but are computationally more expensive. The AFMM focuses on the best of both worlds by producing good results at low computational cost. We explained the theory behind the algorithm and conclude that the algorithm is indeed easy to understand and implement. The best thing about the AFMM is that it only takes one intuitive parameter to set, which even a less-experienced user is able to do. We assess these claims to be correct.

7 Future work

The AFMM may be useful in many more applications, but is not used because it is not well-known. Perhaps implementing the AFMM in a well-known open-source project would increase the popularity.

8 Acknowledgements

The authors would like to thank the reviewers for their useful and thorough comments, which helped to improve the contents, style and structure of the paper.

References

[1] H. Blum. A Transformation for Extracting New Descriptors of Shape. In W. Wathen-Dunn, editor, *Models for the Perception of Speech and Visual Form*, pages 362–380. MIT Press, Cambridge, 1967.

[2] H. Blum and R. N. Nagel. Shape description using weighted symmetric axis features. *Pattern Recognition*, 10(3):167 – 180, 1978. The Proceedings of the IEEE Computer Society Conference.

[3] S. Bouix and K. Siddiqi. Divergence-based medial surfaces. In *Computer Vision - ECCV 2000*, volume 1842 of *Lecture Notes in Computer Science*, pages 603–618. Springer Berlin / Heidelberg, 2000.

[4] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 1992.

[5] R. L. Ogniewicz and O. Kbler. Hierarchic voronoi skeletons, 1995.

[6] R. L. Ogniewicz and R. L. Ogniewicz. Automatic medial axis pruning by mapping characteristics of boundaries evolving under the euclidean geometric heat flow onto voronoi skeletons. Technical report, 1995.

[7] F. Reinders, M. E. D. Jacobson, and F. H. Post. Skeleton graph generation for feature shape description. In *In Joint Eurographics-IEEE TCVG Symposium on Visualization*, pages 73–82. Springer Verlag, 2000.

[8] J. Sethian. A fast marching level set method for monotonically advancing fronts. *Proc. Nat. Acad. Sci.*, vol. 93:1591–1595, 1996.

[9] J. A. Sethian. Fast marching methods. *SIAM Rev.*, 41:199–235, June 1999.

[10] K. Siddiqi, S. Bouix, A. Tannenbaum, and S. W. Zucker. The hamilton-jacobi skeleton. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, pages 828–, Washington, DC, USA, 1999. IEEE Computer Society.

[11] A. Telea and J. J. van Wijk. An augmented fast marching method for computing skeletons and centerlines. 2002.

[12] M. Wan, F. Dachille, and A. Kaufman. Distance-field based skeletons for virtual navigation. In *Proceedings of the conference on Visualization '01*, VIS '01, pages 239–246, Washington, DC, USA, 2001. IEEE Computer Society.

Desired Features of Software Architectural Knowledge Management Tools

Zengyang Li, PhD student of SEARCH Group, University of Groningen

Abstract—Recently, both researchers and industrial practioners are increasingly aware of that architectural knowledge (AK), such as design decisions and rationale, is important for the communication among stakeholders and should be recorded along with architecture design. Since the AK that is created and used during the architecting process is voluminous, broad, complex, and evolving, it is difficult for architects to manually manage AK, thus (semi-) automatical support for AK management is required. Some architectural knowledge management (AKM) tools were developed for this purpose. However, there is no consensus about what features an AKM tool should support. By reviewing selected papers on AKM tools, this paper summarizes a set of knowledge activities and use cases that an ideal AKM tool should support in architecting life-cycle; then, this paper focuses on AK sharing and proposes seven desired features that an AKM tool should possess by considering the characteristics of architecting. Finally, we discuss some questions about the openness level of AK and reputation ranking mechanism of AK and its producers. We argue that this discussion is valuable to the design and development of future AKM tools.

Index Terms— Architectural knowledge, architectural knowledge management, architectural knowledge management tools, tool features

1 Introduction

Software architecture (SA) has been considered of paramount significance in managing the complicated interactions and dependencies between stakeholders [12]. As a reference artefact, SA can help different stakeholders to share knowledge with regard to the design of a software system [1].

In the last decade, most of researchers and industrial practitioners have regarded a software architecture as a high-level design [12] which can be documented using component and connector views [13]. Recently, SA community has been gradually aware of that not only the architecture design itself is crucial to record, but also the knowledge with respect to it [2]. Kruchten et al. defined architectural knowledge (AK) as design decisions plus design, and they focused on the design decisions and their rationale [11]. However, besides decisions and rationale, AK also contains other architecturally significant information. De Boer et al. argued that AK is a collection of entities such as concerns, alternative solutions, decisions, rationale, people, architectural design, processes, and a set of relationships between them [6].

Software architecting is a complex and highly knowledge-intensive process during which a large amount of AK is created and used [4]. Such AK is broad, complex, and evolving and thus difficult to be manually managed by architects. Moreover, owing to the increase in size and complexity of software-intensive systems, diverse stakeholders are involved and architects need to efficiently collaborate with them in the architecting process [4]. Therefore, AK sharing among these stakeholders and across a number of the lifecycle phases is essential and managing AK becomes quite challenging. This situation is intensified in the context of distributed or global software development. Meanwhile, the industry has also been aware of the need for efficient AK sharing within an organization. Therefore, AK needs to be (semi-)automated managed by appropriate tools [4].

Knowledge management has started to play an increasingly important role in SA [9]. Knowledge management comprises a range of strategies and practices used in an organization to identify, create, represent, distribute, and enable adoption of knowledge [16]. Architecture Knowledge Management (AKM) applies the approaches and strategies in knowledge management to SA domain.

Recently, researchers and industrial practitioners developed some AKM tools for different purposes [2][3][4][5][8][9]. These AKM tools place emphasis on different features. For instance, Henttonen et al. studied the open source based tools for sharing and reusing AK with a special focus on "easy to use" feature [5]. Farenhorst et al. proposed an AK sharing portal emphasizing the feature of "Just-In-Time" AK sharing [8]. Some AKM tools, such as ADkwik (Architectural Decision Knowledge Wiki), ADDSS (Architecture Design Decision Support System) and Archium, only support the codification strategy while other tools, e.g., EAGLE and PAKME, support the hybrid strategy [4] (For more details about AKM strategies, please see next section). However, there is no consensus about what features an AKM tool should support. For guiding the development of new AKM tools, a first step is identifying what features an AKM should support. In this paper, by reviewing several scientific papers in the field of AKM, we try to present a set of desired features that an AKM tool should support in architecting process.

The rest of this paper is organized as follows. Section 2 presents the strategies of AKM. Section 3 illustrates a set of knowledge activities that an AKM tool should support. Section 4 distills a set of use cases an AKM should implement. Section 5 outlines the features that an AKM tool needs to support in terms of AK sharing. Section 6 discusses some questions about openness level of AK and reputation ranking mechanism of AK and its producers. Section 7 concludes this paper and outlines the future work directions.

2 AKM Strategies

"Knowledge management codifies and reuses relevant knowledge that is considered valuable in a particular organization" [3]. In recent years, there has been an increasing awareness that it is critical for codifying and managing architectural design and the knowledge pertaining to it for future sharing and reuse in the SA field [2].

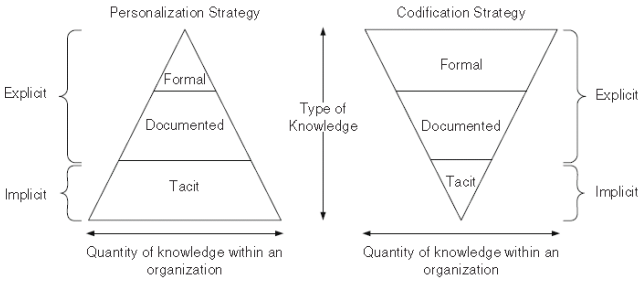


Fig. 1. Pyramid of AK types and the associated management strategies [15]

In knowledge management, knowledge is generally classified into implicit and explicit knowledge [16]. Implicit (or tacit) knowledge resides in people's heads, while explicit knowledge is codified in some form (e.g. a document, or a model). Explicit knowledge can be further classified as documented knowledge (i.e., explicit knowledge documented using natural language or images) and formal knowl-

edge (i.e., explicit knowledge codified representing by a formal language or model of which the exact semantics are defined) [4].

To effectively manage AK, an appropriate strategy needs to be chosen according to application scenarios. There are three knowledge management strategies: codification, personalization, and the hybrid which is a combination of the previous two [4]. Figure 1 shows different knowledge types in the vertical dimension and the differences between codification and personalization strategies in the horizontal dimension.

Codification targets to systematically store AK in order to make the AK available to people in an organization and it emphasizes the significance of formal AK. Personalization focuses on storing information relevant to knowledge sources thereby enabling people to know who knows what and it underlines the importance of tacit AK [2].

3 Desired AK Activities

Tang et al. [3] argue that AK activities that an AKM tool should support can be identified by considering the architecture life-cycle as a set of architecting stages and identifying AK activities in each stage. These AK activities can be regarded as a collection of requirements for development and features of AKM tools.

3.1 Architecture life-cycle

Tang et al. [3] extended the general model of software architecture design proposed by Hofmeister et al. [7]. The extended model indicated that architecture life-cycle includes not only the three stages of architecture design (i.e., architectural analysis, architectural synthesis and architectural evaluation) but also the stages of architecture evolution and maintenance in a system's life-cycle. Figure 2 shows the architecting stages in the architecture life-cycle. The explanations for these stages are as follows:

1. *Architectural analysis* aims to define the problems need to be solved in the architecture. This activity filters and examines architectural concerns and context in order to identify a collection of architecturally significant requirements (ASRs) [7].
2. *Architectural synthesis* proposes architecture solutions to a set of ASRs. By synthesizing existing knowledge, the architect selects the most suitable solutions for the ASRs from a set of available solution choices [7].
3. *Architectural evaluation* ensures that the architectural solutions selected are the right ones. Each candidate architectural solution is measured against the corresponding ASR [7].
4. *Architectural implementation* makes the architecture a detailed design. In this stage, designers and developers need to make more decisions to design and implement the architecture based on the existing one [3].
5. *Architectural maintenance* is the stage in which architectural changes may take place and impact analysis might be done before a new architectural decision is made [3].

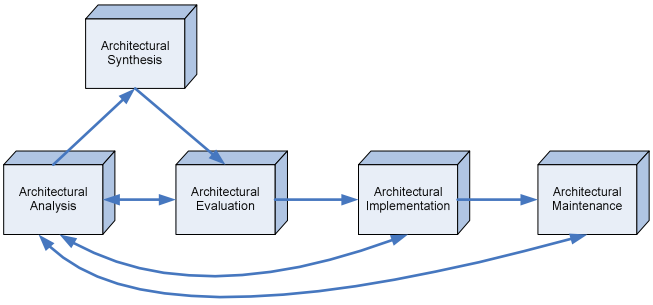


Fig. 2. Architecture life-cycle [3]

AK is mainly produced during the initial three stages, i.e., architectural analysis, synthesis and evaluation. During the architectural implementation and maintenance stages, designers, developers and maintainers use the AK created and captured during the early three stages to support their work. If there are new design issues need to be settled, they may have to revisit the architectural analysis, synthesis, and evaluation stages [3].

3.2 AK Activities

Before identifying the AK activities involved at each stage of the life-cycle of an architecting process, a good understanding of various kinds of AK that may be produced and/or consumed in these stages may be very helpful. To this end, AK can be classified into four types [3]:

(1) *Context knowledge* is a set of information relevant to the problem space, e.g., ASR and the context of a project.

(2) *General knowledge* is a set of knowledge refined as valuable assets that can be drawn on by architects while designing software and systems, e.g., architectural styles and patterns, and tactics.

(3) *Reasoning knowledge* is a set of reasoning information about a design, e.g., design decisions, design rationale, design alternatives, and trade-offs performed.

(4) *Design knowledge* is a set of designs of a system, e.g., components and architectural models.

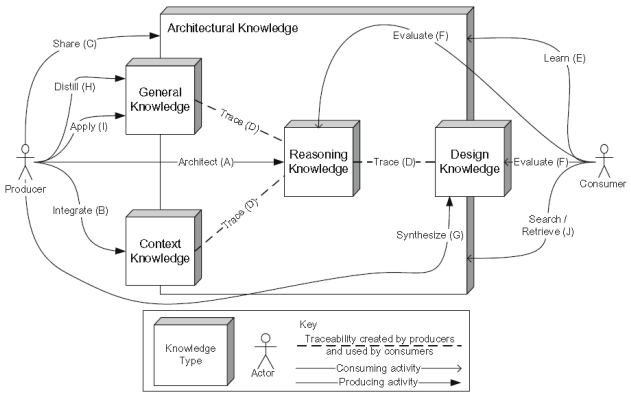


Fig. 3. Architectural knowledge activities [3].

Figure 3 shows the AK activities in the life-cycle of an architecting process. The following describes the AK activities in each stage of an architecture life-cycle.

3.2.1 AK Activities in Architectural Analysis

At the architectural analysis stage, an architect, as a producer, *integrates* context knowledge into already existing AK to identify ASR. Meanwhile, as a consumer, the architect needs to *Learn* and *search/retrieve* the existing AK to make sure if there is any other relevant AK (e.g., previous design decisions) affecting the analysis or not [3].

3.2.2 AK Activities in Architectural Synthesis

At the architectural synthesis stage, an architect, as a producer, needs to propose solutions (a type of reasoning knowledge) for ASRs by *architecting*. To facilitate architecting, the architect may *learn* and *search/retrieve* AK and *apply* general knowledge, such as patterns, to solve the problems at hand. The architect creates a design and *synthesizes* it to capture the design knowledge. Meanwhile, the architect also needs to create the necessary *traces* between the four types of knowledge mentioned above [3].

3.2.3 AK Activities in Architectural Evaluation

During the architectural evaluation stage, an architect *shares* AK

with architecture evaluators to allow them, as AK consumers, to *learn*, *search/retrieve*, and *evaluate* the reasoning knowledge and design knowledge. Furthermore, to efficiently evaluate an architecture, the evaluators often need to *trace* reasoning knowledge to the other three kinds of AK. When an architecture design is evaluated and approved, it may be *distilled* as a general design pattern in general knowledge for future reuse by architects or reviewers [3].

3.2.4 AK Activities in Architectural Implementation

At the stage of architectural implementation, designers need to create a detailed design for an architecture. In this course, they may *synthesize* the design knowledge. To facilitating the designers and developers' understanding of the architecture design for implementation, architects would *share* the AK with them and allow them to *learn*, and *search/retrieve* the available reasoning knowledge [3].

3.2.5 AK Activities in Architectural Maintenance

During the architectural maintenance stage, maintainers would *trace* the design knowledge to the other types of AK to *learn* the rationale of designs and *evaluate* the impact of certain architectural changes [3].

Table 1. AK activities in each architecting stage

Architecting Stage	AK Activities
Analysis	Integrates (B), Learn (E), Search/Retrieve (J)
Synthesis	Architect (A), Trace (D), Learn (E), Synthesize (G), Apply (I), Search/Retrieve (J)
Evaluation	Share (C), Trace (D), Learn (E), Evaluate (F), Distill (H), Search/Retrieve (J)
Implementation	Share (C), Learn (E), Synthesize (G), Search/Retrieve (J)
Maintenance	Trace (D), Learn (E), Evaluate (F)

Therefore, the AK activities involved at each stage of the architecture life-cycle can be summarized in table 1. After integrating those AK activities of the stages in the architecture life-cycle, a full list of AK activities that AKM tools should support include:

1. Architect (A): creating new reasoning knowledge.
2. Integrate (B): integrating context knowledge into AK.
3. Share (C): sharing existing AK of the architecture which the architect is working on with other stakeholders.
4. Trace (D): tracing reasoning knowledge to context knowledge, general knowledge and design knowledge.
5. Learn (E): learning and understanding existing AK.
6. Evaluate (F): evaluating reasoning knowledge and design knowledge to make sure the knowledge is correct.
7. Synthesize (G): using the design decisions and producing the system design.
8. Distill (H): distilling some AK into general AK, such as a design pattern.
9. Apply (I): applying general knowledge using existing solutions (e.g., patterns) to solve the problems at hand.
10. Search/Retrieve (J): searching/retrieving AK for a certain purpose, such as learning.

4 Desired Use Cases

After surveying a series of papers, Liang and Avgeriou [4] formed a panorama set of use cases that can be deemed as the potential set of tool features for AKM. The use cases define the requirements for the development of an ideal AKM tool, i.e. who are the users of it (actors) and with it what would the users do (use cases)?

4.1 Actors

The actors of an AKM tool contain [4]:

1. ***Architects*** who design architectures of software systems may need to transform tacit AK from tacit to documented or formalized knowledge.
2. ***Reviewers*** who are engaged in judging the quality or progress of an architecture.
3. ***Requirements engineers*** who learn AK to facilitate their action in identifying the first-class requirements.
4. ***Developers*** who participate in implementing of the architecture design and decisions.
5. ***Maintainers*** who maintain the system and need to get a good understanding of what impact the decisions they take will exert on the previous architecture.
6. ***Users*** who are the whole collection of system stakeholders of an AKM tool, including the actors mentioned above as specialized ones.

4.2 Use cases

According to the functionality of use case, the use cases (UCs) can be grouped into four categories (see Fig. 4): *consuming AK*, *producing AK*, *knowledge management* and *intelligent support* [4]. Use cases in "consuming AK" enable actors to consume AK for certain purposes; use cases in "producing AK" allow actors to create new and modify existing AK; use cases in "knowledge management" supply general functionality for managing AK data; and use cases in "intelligent support" aim at automating AKM tasks that require either rigor or intelligence.

In figure 4, different colors indicate various types of use cases with different possible actors. The actors of the use cases with light green (i.e., UC5, UC7, UC11, UC13, UC22, UC23 and UC24) are only architects, the actors of the use case with light blue (i.e., UC14) are reviewers, and the possible actors of the remaining use cases are all types of users.

Some use cases can be easily understood since their names are with clear meanings. Some are overlapped with the AK activities illustrated in section 3. For this reason, we just describe the following use cases in detail.

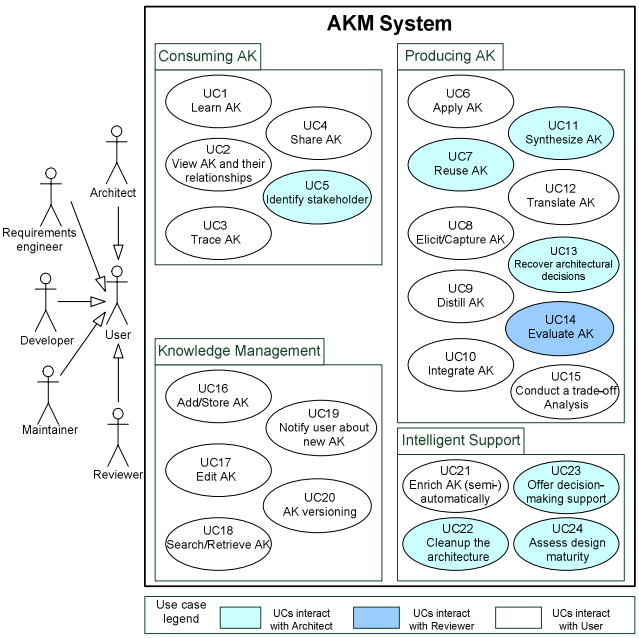


Fig. 4. Panorama of the AKM use case model [4].

- **UC5, Identify stakeholder:** architects identify a certain stakeholder according to specified criteria, e.g., who influences a certain architectural decision most or some architecturally key information should be supplied by whom [17].
- **UC7, Reuse AK:** the architect reuses existing AK from a project in another project context [17].
- **UC8, Elicit/Capture AK:** elicit and capture AK from various project-related resources, e.g. stakeholders and documents [4].
- **UC12, Translate AK:** translate the formal AK based on a certain AK domain model into another domain model to facilitate future reuse [18].
- **UC13, Recover architectural decisions:** the architect rebuilds decisions with their associated rationale from a legacy or 3rd party system [17].
- **UC15, Conduct a trade-off analysis:** analyze the architecture by trading off different quality attributes [17].
- **UC19, Notify user about new AK:** notify the users, who have subscribed to elements on some specific AK topics, about changes to them [9].
- **UC20, AK versioning:** create and manage different versions of various types of AK elements [4].
- **UC21, Enrich AK (semi-) automatically:** create AK content proactively, e.g. automatically interpret then elicit AK from architectural information documented in text without the users' intervention [4].
- **UC22, Cleanup the architecture:** ensure that all the associations between the removed AK and other AK have been removed as well [17].
- **UC23, Offer decision-making support:** supply automated support for the architect in decision-making process [9].
- **UC24, Assess design maturity:** "the architect evaluates when the architecture can be considered as finished, complete, and consistent, e.g. verify whether a system conforming to the architecture can be made or bought" [4].

5 Specific Features for AK Sharing

Section 3 and 4 illustrate the knowledge activities and use cases an ideal AKM tool should support in the whole architecting process. In this section, we focus on the AK sharing and describe what features an AKM tool should have.

Software architecting involves a set of knowledge-intensive tasks. Many different stakeholders located in different sites are involved in these tasks. As illustrated in the introduction of this paper, sharing AK is crucial, especially for reusing best practices, obtaining a more transparent decision making process, tracing between AK elements, and recalling past decisions and their rationale.

Farenhorst et al. [2] defined five main characteristics of architecting based on which they define seven specific features of AKM tools in terms of AK sharing. Subsection 5.1 and 5.2 describe them respectively.

5.1 Characteristics of architecting

The main characteristics of architecting are as follows [2]:

1. ***Architecting is consensus decision making.*** Architecting can be viewed as a decision making process that not only seeks the agreement of most stakeholders, but also resolves or mitigates the objections of the minority to achieve the most agreeable solution.
2. ***Architecting is iterative in nature.*** This iterative nature of architecting is illustrated by the concept of a backlog that is implicitly or explicitly maintained by architects [7]. In this backlog, there are some needs, issues, problems to be tackled and ideas to be used. The architecting workflow is driven by such a backlog. Conceptually, the architecture can not be seen as finished until this backlog is empty.
3. ***Architecting is an art.*** The creativity of the architect plays a significant role in architecting process especially when dealing with novel and unprecedented systems since there may be no codified experience to draw upon. Thus, AK sharing tools should support the architect's creativity instead of constraining it, i.e., methods and tools probably work better if they are more descriptive in nature.
4. ***Architecting impacts the complete life-cycle.*** Architecting affects not only the architecture design phase but also the implementation and maintenance stages. An architecture is never finished before the system's retirement from use, i.e., architecting is performed through the whole system life-cycle. An architecture plays an important role in safeguarding architectural qualities during implementation and maintenance. If relevant AK is not stored correctly knowledge vaporization will happen. Therefore, AK should be available to various stakeholders such as developers and maintainers, instead of only to the architects.
5. ***Architecting is constrained by time.*** In practice, the available time the architects have is usually a heavy constraint on the architecting activities. Often, 'time to market' is a big pressure for the architects and they have to choose for suboptimal solutions.

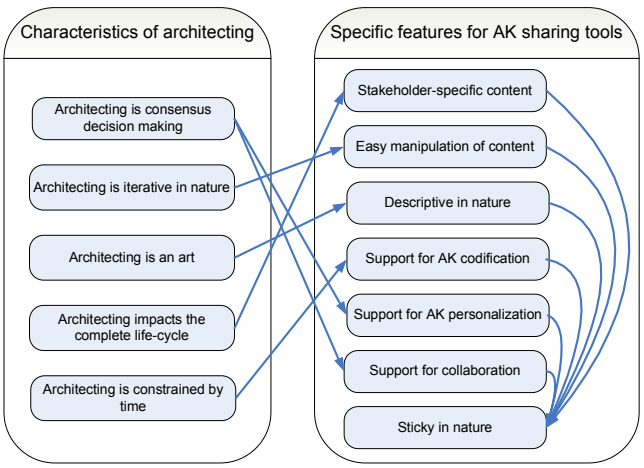


Fig. 5. Specific features for AK sharing tools.

5.2 Specific Features of an AKM tool for AK sharing

Based on the five characteristics of software architecting mentioned above, a set of specific features of an AK sharing tool are proposed (see Fig. 5) as below [2]:

1. ***Stakeholder-specific content***. Because Architecting impacts the complete life-cycle, different stakeholders are involved in the architecting process and they require specialized views on the available AK which they are interested in, such as open issues, approved decisions. AK sharing tools should support to distinguish between certain types of knowledge thus different kinds of users, e.g., developers and designers, can then choose what AK they want to retrieve.
2. ***Easy manipulation of content***. Because architecting is iterative in nature, architects follow a continuous iterative decision making process. Easy manipulation of content can help to accelerate the decision making process, while rigid tool support will just slow it down.
3. ***Descriptive in nature***. Since Architecting is an art, too many restrictions put on the architects will limit their creativities when they are using an AK sharing tool. Therefore, the tools should allow a descriptive perspective on the available AK.
4. ***Support for AK codification***. Because Architecting is constrained by time, architects need support for finding relevant

AK quickly. For some kinds of knowledge that does not tend to change frequently, a codification strategy may be fit best. Then architects can easily retrieve previous solutions that are evaluated and approved, and reuse them.

5. ***Support for AK personalization***. Because Architecting is consensus decision-making, some AK, such as AK about issues under discussion, tends to be evolving and vary often, thus it is not 'stable' enough to codify immediately. For such knowledge, a personalization strategy could be useful to enable architects to find who knows what.
6. ***Support for collaboration***. Because Architecting is consensus decision-making, supporting collaboration between users is an essential feature for AK sharing tools. Since most architects are specialized in certain areas, it makes possible that architects get relevant stakeholders involved in the current decision making activity by collaboration support of AK sharing tools.
7. ***Sticky in nature***. To some sense, this feature can be considered as an orthogonal one to the others since it is an essential property to motivate potential users to start using an AK sharing tool at the beginning. Without this property, users may neglect the tool after having played with it once. Thus, the tool should combine some special elements to keep this tool at a certain level of stickiness. For instance, we can put top N users, who use the tool more frequently and the top N architectural knowledge, which is browsed more frequently, on the homepage of the tool. Through this, we make famous the top N users and the producers of the top N architectural knowledge to the tool users and make the tool attractive to existing and potential users.

6 DISCUSSIONS

In the previous sections, we reviewed the selected papers in terms of the desired features of AKM tools. In this section, we suggest that two additional features, i.e., openness of AK and reputation ranking mechanism, should be considered as supplementary ones of AKM tools besides the features described in section 3, 4 and 5. We believe that this discussion will be beneficial to the development of future AKM tools. We firstly discuss openness of AK, which can be a feature of AKM tools in terms of protecting confidentiality of AK in an organization. In [5], the authors discussed the security of AK, but they did not show how to classify AK and who can access what kind of AK. Thus, there is a need to illustrate this problem in more detail. Then, we discuss reputation ranking mechanism of AK and its producers, which could be a feature of AKM tools specified in facilitating users' understanding of AK quality. Moh and Kaul [19] mentioned that user reputation should be considered when building a tool for the visualization of AK. The aim is to help user understand how reliable the supplied content is. However, the authors did not give the details to form such a ranking mechanism. Besides, we also advise that not only the user reputation but also the AK reputation need to be measured in an AKM tool.

6.1 Openness of AK

With our work experience in industry, usually, for reasons of confidentiality, some key AK in a specified project is not allowed to be published outside. Because this kind of AK is relevant to the core of the system and it is only accessible to members of this project. Some AK, such as context knowledge, just makes sense in the project context where the knowledge comes from. Therefore, such knowledge does not need to be published outside the project. Generally, there is some AK valuable to the organization in a project since the organization usually is possible to develop similar systems in the future. Thus, such AK should be open to the organization. Some AK that is evaluated and approved may be as universal knowledge, such as design patterns and architecture patterns, for the software community. Thus, as shown in figure 6, we define three openness levels for AK of a project.

Project-specific. Project-specific AK can be shared among the stakeholders within a particular project. Such AK includes the key AK which is not allowed to be open to the outside of the project and the AK which makes sense just with in the project. The quantity of this kind of AK is the most among the three levels of AK.

Organization-specific. Organization-specific AK can be shared within the whole organization in which the members can learn this kind of AK of the project and reuse it in other projects. This kind of AK usually contains the AK which is valuable for the similar kind of projects of the similar businesses in the organization. Also, some AK special relevant to the organizationally architectural principles (e.g., projects in this organization must use web services instead of ".net Remoting" technique since the organization has developed a web services library in the business domain) should be included into organization-specific AK.

Public. Public AK is open to all potential users or consumers. Such AK is with high reusability and may be used in all software projects. The quantity of public AK is the least among the three levels of AK.

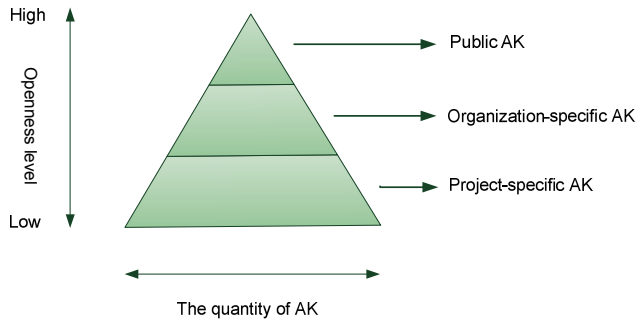


Fig. 6. The openness model of AK.

This openness model classifies the AK that created in the same project into 3 types from the perspective of accessibility, and it is a different classification approach from the one in section 3.2, which groups the AK both consumed and produced in a certain project into 4 categories: context, general, reasoning and design knowledge. Apparently, there is intersection between different types of AK from the two classifications. For example, general AK is also public AK; reasoning AK generally is project-specific and organization-specific AK.

An example of the application of this openness model is following. We worked in a large company that is one of the top 5 telecom equipment and services provider in the world. The main task of our department is developing automated testing tools for testing the hardware function of various series of base stations. The hardware includes digital parts and radio frequency parts. There is another department C with similar tasks to our department, but that department's target test object is high-performance routers that only have digital parts. Thus, some AK about digital parts test in the project of our department is useful to department C and should be organization-specific. The AK on radio frequency parts test just applicable in the projects of our department and therefore it should be project-specific. The AK about digital parts test can be distilled as a pattern that will be beneficial to the domain of digital parts test, such as computer mainboard test. Thus, the distilled AK can be public to all potential users in this field.

6.2 Reputation ranking mechanism

The quality of AK makes large impact on the use and sharing of AK. Because architecture can strongly affect a software-intensive system in the whole life-cycle, AK with poor quality will cost users much effort to validate the correctness and may result in negative effects upon the maintenance and evolution of the system architecture.

To help users to get a good understanding of the quality of AK, we may create a reputation ranking mechanism to rank AK and AK

producers. The main idea is: (1) We calculate the reputation of an AK element according to the scores assigned by the AK element's users; (2) Different users of the AK element have various reputation values, and the score assigned by the user with higher reputation should have a higher weight on the calculation of the AK element's reputation value; (3) The reputation value of an AK producer is the average reputation value of all AK elements of the producer.

For example, An AKM tool offers a set of criteria for appraising AK and then users can give a score to AK according to the criteria and comment it. We assume that there is an AK element k and the number of its users is n_k. The set of reputation values of its users is $\{ur_1, ur_2, ..., ur_{n_k}\}$, and the corresponding set of scores assigned by the users is $\{s_1, s_2, ..., s_{n_k}\}$. We define the reputation of k, akr_k, as

$$akr_k = (\sum_{j=1}^{n_k} ur_j \times s_j) / (\sum_{i=1}^{n_k} ur_i) \quad (1)$$

We take the reputation of the users of the AK element into account in this definition in order to increase the weight of the score assigned by the user with higher reputation and decrease the weight of the score assigned by the user with lower reputation. When users search or retrieve AK relevant to a particular topic, results are listed with the reputation values so that users can easily judge the quality of AK and select the target AK more effectively. Furthermore, the reputation of AK producers may influence the acceptance of the AK created by them. People tend to believe that the AK produced by a producer with higher reputation value is more possible to have a good quality. Therefore, we can use the average reputation value of all AK produced by a producer as a reputation measurement associated to the producer. When users search the AK about some certain topic, they can easily pick out the AK whose owner is with higher reputation.

7 Conclusion

As AKM becomes critical to strengthening an organization's architectural capabilities, tooling support for this management is required. To facilitate managing AK, some AKM tools were developed for this purpose. However, there is not a consensus about what features an architectural knowledge management tool should support.

By reviewing selected papers on AKM tools, this paper summarizes a set of AK activities and use cases that an ideal AKM tool should support in the architecture life-cycle. Then, we focus on the AK sharing and illustrate seven specific features an AKM tool should possess by considering the characteristics of architecting. Finally, we propose an openness model of AK, which suggests the AK in an organization should be distinguished to 3 levels, i.e., project-specified, organization-specified and public AK. And we also suggest that an AKM tool should support a reputation ranking mechanism of AK and its producers. We believe that all these AK activities, use cases, specific features for AK sharing and the model of AK openness and the reputation ranking mechanism make up the most important desired features of AKM tools.

In this paper, many features are conceptually described. However, the technical details of the features have not been discussed yet, e.g., what steps should we take to implement a use case? Is there any issue remained to be resolved in each use case? What technologies should we employ to implement each feature? How to make a trade-off among different features when using a specified technology? In the next step, we will investigate these questions and generate a guideline for future AKM tool development.

Acknowledgements

This work was supported in part by a grant from FNR, Luxembourg. The author would like to thank Dan Tofan, Sjoerd Hemminga, Fernand Geertsema and Peng Liang for their reviewing of and valuable suggestions to the paper.

References

[1] P. Avgeriou, P. Kruchten, P. Lago, P. Grisham and D. Perry, "Architectural Knowledge and Rationale – Issues, Trends, Challenges", ACM SIGSOFT Software Engineering Notes, Vol. 32 Number 4, pp 41-46, Jul. 2007.

[2] R. Farenhorst, P. Lago and H.V. Vliet, "Effective tool support for architectural knowledge sharing". In: First European Conference on Software Architecture (ECSA'07), pp. 123–138, 2007.

[3] A. Tang,P. Avgeriou, A. Jansen, R. Capilla and M. A. Babar, "A comparative study of architecture knowledge management tools", The Journal of Systems and Software 83 (2010) 352–370, 2010.

[4] P. Liang and P. Avgeriou, "Tools and Technologies for Architecture Knowledge Management", In Software Architecture Knowledge Management: Theory and Practice, pages 91–111. Springer, 2009.

[5] K. Henttonen and M. Matinlassi, "Open Source Based Tools for Sharing and Reuse of Software Architectural Knowledge", Joint Working IEEE/IFIP Conference on Software Architecture, 2009 & European Conference on Software Architecture. WICSA/ECSA 2009. Oct. 2009.

[6] R.C. de Boer, R. Farenhorst, P. Lago, H. van Vliet, V. Clerc and A. Jansen, "Architectural knowledge: getting to the core". In: Third International Conference on the Quality of Software Architectures (QoSA), 2007.

[7] C. Hofmeister, P. Kruchten, R.L. Nord, H. Obbink, A. Ran and P. America, "Generalizing a model of software architecture design from five industrial approaches". In: Fifth Working IEEE/IFIP Conference on Software Architecture (WICSA 2005), pp. 77–88, 2005.

[8] R. C. de Boer and R. Farenhorst, "In Search of 'Architectural Knowledge'". Proceedings of the 3rd international workshop on Sharing and reusing architectural knowledge (SHARK'08), pp. 71-78, 2008.

[9] R. Farenhorst, R. Izaks, P. Lago and H. van Vliet, "A Just-In-Time Architectural Knowledge Sharing Portal". Seventh Working IEEE/IFIP Conference on Software Architecture (WICSA 2008), pp. 125-134, 2008.

[10] A. Nour, B. Sarah and M. Ivan, "Architectural knowledge management in global software development: a review". 2010 5th IEEE International Conference on Global Software Engineering (ICGSE 2010), pp. 55-63, 2010.

[11] P. Kruchten, P. Lago and H. van Vliet, "Building up and Reasoning about Architectural Knowledge". Quality of Software Architecture (QoSA), pp. 43-58, 2006.

[12] L. Bass, P. Clements and R. Kazman, "Software Architecture in Practice", 2nd edn. SEI Series in Software Engineering. Addison-Wesley Pearson Education, Boston, 2003.

[13] J. Ivers, P. Clements, D. Garlan, R. Nord, B. Schmer and J. R. O. Silva, "Documenting Component and Connector Views with UML 2.0", TECHNICAL REPORT, University of Carnegie Mellon/Software Engineering Institute, 2004.

[14] P. Kruchten, P. Lago, H. van Vliet and T. Wolf, "Building up and exploiting architectural knowledge". In: Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture (WICSA), pages 291-292, 2005.

[15] A. Jansen, "Architectural design decisions", PhD thesis, University of Groningen , 2008.

[16] I. Nonaka and H. Takeuchi, "The Knowledge-creating Company: How Japanese Companies Create the Dynamics of Innovation". Oxford University Press Inc, USA, 1995.

[17] J. van der Ven, A. Jansen, P. Avgeriou and D. Hammer: Using architectural decisions. In: Proceedings of the 2nd International Conference on the Quality of Software-Architectures (QoSA), LNCS, vol. 4214, pp. 1–10, 2006.

[18] P. Liang, A. Jansen and P. Avgeriou, "Collaborative Software Architecting through Knowledge Sharing". In Collaborative Software Engineering, pages 343–368, 2010.

[19] T.S. Moh and A. Kaul,"A Prototype for Visualized Architecture Knowledge Collaboration Services".2010 IEEE International Conference on Granular Computing, 2010.

Wikis Support in Architectural Knowledge Management for Sharing and Reuse: the WikiPL Approach

Konstantinos Tselios and Manuel Martiarena

Abstract—The amount of architectural knowledge that is produced, consumed, re-produced and consumed again in the development of a medium-large software project is massive. As a project grows, complexity increases it and becomes impossible for the software architect to manually manage the situation. This knowledge needs to be shared and reused between many stakeholders. This study finds wikis appropriate to achieve efficiency in Sharing and Reuse of Software Architectural Knowledge (SHARK). We combined key characteristics and evaluation frameworks' criteria, extracted from previous experiences. This literature review leads us to map the key functionalities of the wikis and the desired properties of the SHARK tools. Lastly we present the WikiPL approach and review and discuss its functionalities, with respect to our theoretical findings. WikiPL addresses almost all the desired SHARK properties and it provides a reusable way for preserving architectural designs using unit tests. The design can be formulated into Python code and it can also be documented and shared within a collaborative wiki environment. We successfully simulated small-scaled architectural designs using the WikiPL environment; this study provides the details and the results of the simulation. In overall we argue that wiki tools can be used to support and facilitate SHARK and we propose using WikiPL as a tool for preserving the validity of the shared knowledge.

Index Terms—Software Architecture, Architectural Knowledge Management Tools, Knowledge Sharing and Reuse, SHARK, Wiki, WikiPL.

1 Introduction

Software architecture is a discipline that is rapidly growing the last few years [8]. There are numerous software design techniques and methodologies used over the years, but this knowledge was only lying inside the minds of the architects. The need for feedback from a community that supports and comments on their work led to investigation into previously unknown knowledge domains and expertise. Architects started to exchange experiences, techniques and tactics.

Sharing and reuse of knowledge constitute an effective way to improve software architecture. Nevertheless, it could be very difficult to provide the architects community an effective tool for submitting, viewing and searching contributed knowledge. An effective tool should cover the needs of its participant and it should also be time-tolerant in order to gather and preserve knowledge. Wiki tools are web-based content management tools [1]. Their content could be seen and manipulated by multiple authors at the same time. This study reviews and discusses wiki functionalities [2][3][16] and desired properties [8][9] regarding SHARK. Moreover we are interested in preserving the validity of the shared knowledge. The WikiPL tool is examined, as it is a Python programming environment in a wiki [15], which uses unit testing for preserving the validity of its content over subsequent changes.

Chapter 2 provides definitions of basic concepts of software architecture and architectural knowledge and its management. We provide an insight of the need for knowledge management in the software architecture practice. Wikis are presented in chapter 3 and their key functionalities are addressed to desired properties for SHARK tools. The approach of WikiPL and our experience using it as a SHARK tool are discussed in chapter 4 and chapter 5 reviews the approach with respect to theoretical mapping of chapter 3. In chapter 6 we discuss this study's findings and future work and finally in chapter 7 we provide conclusions.

- *Konstantinos Tselios @ University of Groningen E-Mail: k.tselios@student.rug.nl*
- *Manuel Martiarena @ University of Groningen E-Mail: m.martiarena@student.rug.nl*

2 Software Architecture

Software architecture as a concept was identified in the early seventies, when scientists realized that the structure of a software system is important; it increases the chances of getting a good product. In the last decade it was mainly considered a high level design, a set of components and its relations, represented in different views [8].

There is not a single definition of Software Architecture. We consider the definition provided in [26] to better fit in our literature review. We provide its outline in this section. Software architecture is a representation of a software system in an architectural language. The architecture should display a clear separation of concerns in the observable and the non-observable behaviours of the system. It should support both static and dynamic components through a set of well-documented interfaces or contracts known to the clients. It should be described in a set of patterns to facilitate clarity and serve as a simple mean of communication between the system's stakeholders. In overall, software architecture should support changeability, expandability and reusability amongst other characteristics. Roughly, we conceptualize software architecture as the organization of interacting software components, acting as a bridge between requirements, engineering and system design.

2.1 Software Architecture Trends

The amount of architecting information of a medium/large software development project grows and becomes massive, complexity increases and becomes impossible for the architect to manually manage this situation. Many stakeholders participate in the decision-making process and they are often in different physical locations. In software architecture the following trends are perceived [14]: (a) Increased collaboration: large groups of stakeholders from different background and expertise participate in the discussions of architectural issues. In order to collaborate, they need to have access to the information in an understandable way and in a level of complexity in accordance to the task; (b) Focus on decision making: many decisions that take place in architecting processes are decisions that matter, and have a large impact on the final system; (c) Distributed development: In the development of a medium-large software project, there are many groups and architects located in

different offices, departments, buildings, countries (Global Software Development [3]); and (d) Need for reusable assets: The rapid progress of software architecture implies continuous evolution of solutions. Reusable assets can prevent architects from re-inventing solutions.

2.2 Architectural Knowledge

If software architecture is conceived as the bridge between requirements, engineering and system design, architectural knowledge could be seen, amongst others, as the rationale on how to build it [6]. Not only architecture design is important to be captured in documents, but also the knowledge attached to it. If this knowledge is not stored or documented it disappears, leading to additional costs in maintenance and inconsistencies in design over time.

Additional reasons to share architectural knowledge are [7]: (a) Encourage reuse: contribution to the community by sharing architectural styles, patterns and experiences with other participants, makes knowledge available to reuse; (b) Facilitate learning: the ability to extract information about the rational of decision-making facilitates in understanding the solutions in a shorter period of time; and (c) Promote collaboration: sharing architectural knowledge allows finding rationales for decisions and access important information sources like artefacts or people.

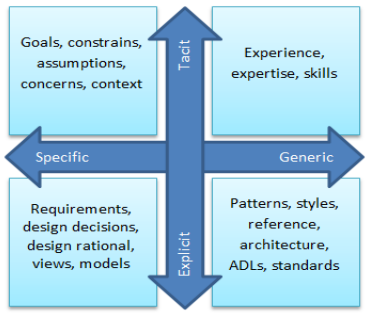


Figure 1: Specific and generic architectural knowledge [19]

There are two types of architectural knowledge, the specific and the generic [22]. Specific architectural knowledge is the sequence of design decisions, including the rationale in their combination to provide the architectural design. Specific architectural knowledge can be modelled and presented to the stakeholders in different ways, such as different types of UML diagrams, scenarios and use cases [23]. Generic architectural knowledge is often tacit knowledge, lying inside the heads of the architects, formed by experience, domain knowledge or expertise. It includes styles, patterns, tactics and experience in using specific technologies, tools and methods. This kind of knowledge is hard to document and model and it is the most valuable asset of an organization. The major problem regarding the capital of knowledge is that it has free will and walks home every day [10].

Figure 1 depicts different kinds of architectural knowledge, organized in four quarters: from Specific-Explicit knowledge, more easy to model and store, to the Tacit-Generic knowledge, residing only in the minds of people which makes it difficult to store and share [19].

2.3 Architectural Knowledge Management

Architectural Knowledge (AK) in software engineering is growing more diverse and vast. There is a greater need to improve the way in which this knowledge is stored and shared [10]. There are several factors that make architectural knowledge sharing and reuse (SHARK) difficult, i.e., documenting knowledge is time consuming and contributors usually do not want to invest time on it. Other perspectives state that knowledge is highly valuable, thus it should not be shared.

There are two defined approaches to Knowledge Management strategies [25]: (1) Personalization relies upon the tacit and implicit knowledge and is more focused in sharing based on the relation of people. The knowledge to transmit includes expertise, ways of thinking and analytical advices amongst others. Personalization could be simply described as a "people-to-people" approach. (2) Codification is based on the explicit knowledge. This strategy approach is more focused on the use of technology to enable, store, retrieve and reuse of explicit knowledge, emphasizing on documenting work processes, best practices and guides. Codification can be described as a "people-to-document" approach. It is important to understand that in AK management it is necessary to provide a balance between the two strategies. AK management tools should support a hybrid approach to stimulate its use in organizations.

3 Wikis

Understanding the terms of architectural knowledge and sharing and reuse of architectural knowledge, we are able to discuss tools that support and facilitate these domains. This chapter provides a brief flashback to the history of the Wikis. Further on, we describe key wiki functionalities and desired SHARK tool properties as defined in literature, and provide a mapping between them. We close the chapter providing a view of how wikis support architectural knowledge sharing and reuse.

3.1 Background

The "wiki wiki" word stands for "quick" in Hawaiian [13]. The first WikiWikiWeb was developed in 1994 by Cunningham to serve as a collaborative software running in a web environment [1]. The main scope of a wiki system is contribution and exchange of knowledge, allowing collaboration amongst the participants. As a kind of Web-in-the-Web system [5], the pages of the wiki are interlinked and content can be edited by multiple authors using a simplified markup language or a what-you-see-is-what-you-get text editor. Basically, it is a lightweight web-based content management application, offering an ease of interaction and collaboration of many authors.

3.2 Wikis Support in Knowledge Sharing and Reuse

A well-structured content, supported by a strong collaborative community that keeps it up-to-date and an effective searching mechanism, make wikis a tool that can strongly support architectural knowledge sharing in an effective way. Reuse of knowledge is more effectively performed if the collection of information, from different knowledge sources, is not a complicated and difficult procedure. In other words, we argue that reuse of knowledge requires an ease of *extracting* the information that one needs.

When someone performs a search on the Web to retrieve a topic-specific piece of information, one can be found in front of an ocean of relevant information; a large amount of previously contributed knowledge. We argue that relevance does not necessarily imply usefulness as well. This simply means that initially relevant information served to a user, as a search result, is not always referenced to the user's interest; thus this piece of information, or knowledge contribution, is useless. Wikis answer the question -"*How does this specific knowledge contribution supports one's decision to reuse it?*". This is also the main concern of software architects with respect to AK reuse. They need a tool to facilitate their design decisions, based on existing concepts in order to avoid reinventing the wheel [27].

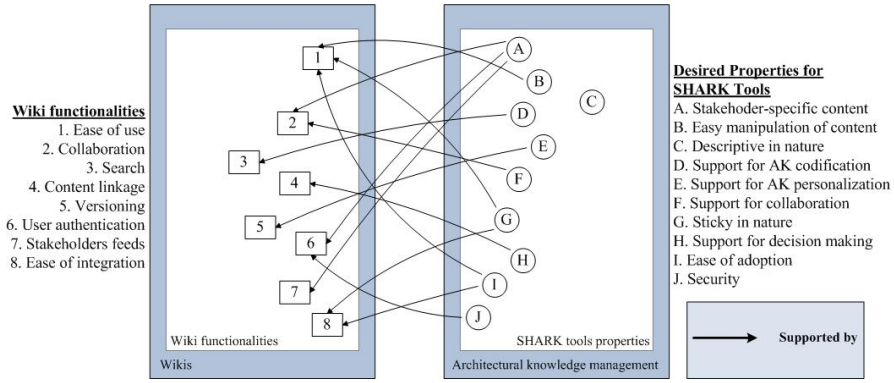


Figure 2: Mapping of Wiki features to desired properties of SHARK tools

It is common conception that concurrent edits performed by numerous participants rarely lead to structured, well-defined and comprehensive content. However, wiki software offers an evolutionary process where edits which are coherent tend to survive subsequent changes and corrections. This model ensures that only qualitative edits remain and the overall quality of an article increases over time. Therefore, wiki serves the user useful and valid information.

3.3 Wiki Functionalities

Cunningham [2] defined the design principles of wikis, whereas Clerc *et al*. [3] defined important generic wiki functionalities. Tang *et al*. [16] set certain criteria in order to perform a comparative study for SHARK tools. The important key characteristics we extracted from the aforementioned sources are:

1. **Ease of use** – an everyday user can easily view and edit wiki web content. Sometimes though, a user needs to hold some limited experience to perform editing and linkage using markup language [3].
2. **Collaboration** – wiki content is dynamic and users can navigate through multiple versions of wiki pages. Hence, many authors can edit content while the tool performs versioning. In this way "wiki users are *prosumers*", as stated by Ellie Rennie [4]; they produce by editing content [-*pro*ducers] and they consume by viewing it [-con*sumers*] at the same time.
3. **Search** – users can search in wiki content requesting an article title or by performing full-text search.
4. **Content linkage** – pages are interlinked within a wiki environment. A wide variety of articles having links back and forth to each other may be inconvenient for someone to find one's way through this content [3]. Reinhold [5] proposed the usage of wiki trails to better suit the knowledge requirements of the environment and its stakeholders.
5. **Versioning** – the wiki content is not static; it may rapidly change and grow large by the participant's potential contribution of knowledge. The versioning mechanism enables back tracking in previous versions of the same content, meaning that the status of the wiki can revert at any time.
6. **User authentication** – authentication assures user identification. Content versioning, also documents the name of the authenticated author; this makes the authors traceable with respect to their activity and/or knowledge domain. Moreover, authentication provides administrations to perform authorization, by limiting users' rights in editing specific wiki content.
7. **Stakeholders feeds** – sharing new content with others in a wiki community enables mechanisms to use RSS-feeds for pushing relevant knowledge content to targeted stakeholders.
8. **Ease of integration** – the wiki environment is open to integration with other tools that can provide extra functionality. Several approaches of wikis exist integrating a variety of external plug-ins to facilitate the usability of the tool itself.

3.4 Tools Support in SHARK

In this study, we extracted desired properties based on the needs of architects, as defined from Farenhost *et al*. [8] and a proposed evaluation framework defining criteria for selecting a SHARK tool for organizational use, developed by Henttonen and Matinlassi [9]. The characteristics of architecting are reflected in the extracted properties.

A. **Stakeholder-specific content** – a software project has many interfering stakeholders during its entire lifecycle. A clear distinction between knowledge committed by different stakeholders is an important aspect that a tool should cover. This enables the user to extract the desired knowledge reflecting a group of profiles of users; i.e., architects, developers or managers.

B. **Easy manipulation of content** – the architectural process is iterative by nature. Architects make and verify decisions in a continuous iterative flow. The ease of content manipulation that a tool provides, can speed up the decision making process in software architecture.

C. **Descriptive in nature** – there should be freedom of choice in the decision making process. The environment should not provide strict and guided modelling and hence, should allow architects to unfold their creativity.

D. **Support for architectural knowledge codification** – the tool should provide the architects with efficient and quick searching of content that is not frequently changing. This content constitutes already proven solutions that the tool should provide a codification strategy quick access for its retrieval.

E. **Support for architectural knowledge personalization** – a personalization strategy can be useful to architects when content is rapidly changing. The tool should offer traceability of authors in a certain domain of architectural knowledge.

F. **Support for collaboration** – collaboration of different stakeholders facilitates the decision making process. Furthermore, it enables discussion and negotiation regarding specific content in several domains of

expertise. An important positive impact of collaboration is that software development can be more effectively divided and managed.

G. **Sticky in nature** – the stickiness of the tool refers its ability to attract the users and provide comfortability. This ability is also facilitated by integrating special features that motivates the user to adopt it as one's primary method for architectural knowledge management.

H. **Support for decision-making** – the knowledge contributes should support its coherence and its usefulness in order to persuade the architects to reuse it. This could be facilitated by provision of additionally documentation of discarded design decisions.

I. **Ease of adoption** – the tool should be easily integrated in the stakeholders' decision-making process. Customizability of the tool plays an important role in accommodating organizational needs. Thus, a tool should be flexible to serve the needs of its environment and its stakeholders.

J. **Security** – the tool should guarantee that knowledge contributions are coherent and accurate in order to preserve the validity of the contribution.

3.5 Mapping Wiki Functionalities to Desired Properties for SHARK Tools

Clerc *et al*. [3] discussed a mapping method, which we followed as a guideline in our work. The mapping Clerc *et al*. performed resulted in assigning wiki functionalities with AK management in Global Software Development (GSD). In their study they "focus on using AK effectively to overcome the challenges associated with GSD practices" [3], by first identifying the wiki functionalities and best GSD practices for AK management, and concluding on how wiki functionalities can be used to implement these practices. In this study we first extracted some wiki functionalities referred in [3]–with additions from other studies [2][16]–and software architects' needs for SHARK tools [8][9], and we mapped them to conclude that wikis support sharing and reuse of AK.

The key functionalities of the wikis defined in §3.3, can be mapped to the requirements of the software architects mentioned in §3.4, regarding SHARK. The upper left of figure 2 lists the eight wiki functionalities and the upper right side lists the ten desired properties for SHARK tools. The mapping of the lists is depicted below in the same figure and shows the relation between two elements pointing each other using arrows. The arrows show that one or more wiki functionalities support one or more desired SHARK properties.

Stakeholder-specific content needs the users to act as authenticated prosumers in order to contribute and consume knowledge in a clearly distinguishing and traceable way. In addition, stakeholders' subscriptions feed them with relevant knowledge content. One of the main functionalities of the wikis is ease of use and manipulation of content, where authors should have none, or limited experience in markup language usage. The codification strategy urges the support of an effective searching mechanism, whereas the personalization strategy proves useful when supported by versioning. Wikis support both personalization as well as codification [17]. In order to be "sticky" a tool should provide extra functionality and also should cover the needs of a wide variety of potential users. Interconnected content supports the decision making process, and user authentication and versioning increase the security of the content. The ease of use and integration causes adopting a tool more easily in a process or an organization. Lastly, property C, as we see in figure 3, is not addressed to a wiki functionality. That happens because a tool providing content management capabilities, gives its user the opportunity to decide whether to respect its content as a barrier to the user's creativity, or as a provision of free choice.

4 WikiPL Approach and Experience

After providing a view on *why* is a wiki tool appropriate to support SHARK, we discuss *how* a tool actually does what we claimed in the previous chapter. We examine the WikiPL approach and we discuss how it can be used to support SHARK. Further on we map WikiPL's functionalities with respect to our findings on §3.3 to desired properties of SHARK tools presented in §3.4.

4.1 The WikiPL Approach

WikiPL is a programming environment in a wiki. Each article can be a function, a class or any other piece of code in the Python programming language [18]. WikiPL joins developers with different backgrounds into collaboration, under a common programming environment without strict central administration. Each developer contributes programming content in the area of one's own expertise. The implementation of WikiPL is based on the MediaWiki content management system, which is well known by its implementation on the Wikipedia project [1].

In a WikiPL article, each section has a different functionality [15]. The "Code" section is where the source code of the article resides. When making an edit to this section, the code is checked for syntax errors and a unit tests check is enacted. The edit is saved if all the checks succeed. "Unit tests" are small snippets of code that verify the integrity and satisfiability of another piece of code. Another special section is the "Parameter" section, where a user can define the parameters of the function that is hosted in an article and its definition is converted to HTML form elements [15][18]. In WikiPL, a method can be executed within an article in three different ways: (a) by local downloading a bundle with the source code, the calling functions and classes, and the running parameter; (b) by copying this bundle in a remote server, i.e., Amazon EC2; or (c) by converting the Python code into JavaScript that is subsequently executed in the user's browser [15]. A button for each execution method is provided. The final special section is the "Permissions" that contains lists of users with the edit permissions to the code, unit test and documentation sections. WikiPL offers a Python library that enables the downloading of the code, suited in a WikiPL article, into a local Python namespace.

4.2 The WikiPL Experience

In its current form and use, WikiPL does not support architectural knowledge but development knowledge. We experimented and tried to treat this tool as a tool for SHARK that could be used in an organization environment. All our actions were performed in a sandbox of the existing implementation of WikiPL, thus the results cannot be publically visualized.

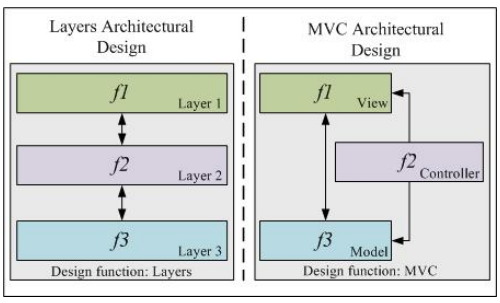


Figure 3: The layers and MVC architectural designs

Instead of adding articles that are pieces of Python code, we created articles that document the relationship of different articles, the order of their execution and their arguments. In this way we simulated an architectural design concept in a small scale. These articles are further on referred as design functions. The concepts we committed to WikiPL were based on two known software pattern designs; the layers pattern and the model-view-controller (MVC) pattern [24].

Figure 3 shows the article functions *f1*, *f2* and *f3*, and their possible architectural designs. The layered architecture imposes that bidirectional communication between *f1* and *f3* is only performed through *f2*. As an alternative, the MVC architecture imposes bidirectional communication to be only performed between *f1* and *f3*, while *f2* only sends data to *f1* and *f3*. The preservation of the communication architecture of the layers and MVC designs is performed by unit tests. These unit tests are performed in the article functions so as to satisfy the conditions of the relevant design function. This means that if i.e. function *f1* calls function *f3*, the *f1* unit test for the layers design function, will not be satisfied.

For composing the code of a design function we used dependency check functions, already provided by WikiPL. These return which functions are being called from another function (`Get_links_from_wikipl_article`) and which functions this specific function calls (`Get_links_to_wikipl_article`). In this way we set design conditions that must return `True` in order to be valid. The code of the `design_layers` function is provided below:

```
def Sandbox_design_layers(f1, f2, f3):
   conditions = []

   #f1 is called from f2 and calls f2
   conditions += [f2 in Get_links_from_wikipl_article(f1)]
   conditions += [f2 in Get_links_to_wikipl_article(f1)]

   #f2 is called from f1 & f3 and calls f1 & f3
   conditions += [f1 in Get_links_from_wikipl_article(f2)]
   conditions += [f3 in Get_links_from_wikipl_article(f2)]
   conditions += [f1 in Get_links_to_wikipl_article(f2)]
   conditions += [f3 in Get_links_to_wikipl_article(f2)]

   #f3 is called from f2 and calls f2
   conditions += [f2 in Get_links_from_wikipl_article(f3)]
   conditions += [f2 in Get_links_to_wikipl_article(f3)]

return reduce(lambda x,y: x and y, conditions)
```

Code 1: The design function Layers in WikiPL.

For composing the unit tests in article functions we considered which conditions should be satisfied. In the layers architectural design, i.e. *f1*, *f2* and *f3* unit tests should satisfy (return `True`) the `design_layers` function conditions. These conditions impose the communication architectural design of the layers pattern, which we described earlier. We noticed that the unit tests have large execution time, as they need to check the satisfiability of multiple functions. The unit test code of *f1*, *f2* and *f3* that satisfies the `design_layers` function is provided below:

```
def test_for_architectural_design():
   if not design_layers(f1, f2, f3):
      return "function failed to satisfy the architectural
              design conditions"

      return True
```

Code 2: The unit test of function f1 in WikiPL.

In the documentation area a rationale of the decision of each design is composed and the design functions and their relationships are illustrated as blocks and arrows. The design function can be executed and it can provide results that can justify or not the initial conceptual rationale; i.e., high performance is required, thus the design function should have shorter execution time than an alternative one. There is also a sample unit test code provided that the article functions should include. Furthermore, in the "See also" section of the article we provided links with similar or alternative architectural designs.

5 RESULTS

So far we discussed about the approach of WikiPL and our experience using it as a tool for SHARK. In this section we provide the results of our study; identification of WikiPL's functionalitites and the mapping between these functionalities and desired properties of tools for SHARK.

5.1 Reviewing the WikiPL Approach

The approach of the WikiPL implementation is not semantic but we identify this as a non-negative functionality. We provide the rationale of this argument in this section that reflects our literature review.

The semantic wikis combine the typical wiki functionality adding meaning to the content. As Souzis stated [20], even for experienced users, writing the required precise statements for achieving a semantic concept is much more time consuming than writing informal text. For having coherent, consistent and structured semantic information, the ontology evolution and convergence should be left to advanced tools in the hands of experts [21]. Provision of contextual presentation, improved navigation, semantic search and reasoning [11][12] may be able to efficiently facilitate SHARK in the near future, when advanced tools will prove fully functional. On the other hand, it is very important for a wiki, like WikiPL, to have a semantic infrastructure lying underneath its surface. We do not provide any further information on semantic wikis because it is outside the scope of this study.

5.2 Mapping WikiPL functionalities to Desired Properties of SHARK tools

WikiPL uses the typical and well-known web-based approach of a wiki. In this way it provides the user with comfort and increases the learning time of the tool. In addition, it urges software development decentralization; a participant may add the documentation and the desired unit tests and "outsource" the development to a third-party while at the same time ensures the validity of a potential code contribution. Besides viewing the content, the user in this approach, actually has the capability of executing it in the environment itself.

Using namespaces for separating different groups of stakeholders is a functionality that WikiPL also provides. It is required though, to have limited programming experience on the Python language for a user to compose the code section of an article, so the manipulation of the content cannot be considered as easy. The AK codification is partly supported, as the users can document their rationale but semantic infrastructure is not used. Searching for already proven solutions in the current implementation does not constitute a WikiPL functionality. Personalization on the other hand is supported, as WikiPL offers traceability of authors and versioning. As every wiki tool urges collaboration [2], so does the WikiPL. A discussion area exists for participants to commit their comments and interact in an asynchronous way. While providing certain permissions to users, or groups of users, the collaboration may be restrictive but this preserves the validity of the content. In the documentation area one can provide details on the committed AK, while the code can be executed, if applicable. In this way one can justify the decision of designing and using this specific piece of AK.

Desired properties of SHARK tools	A	B	D	E	F	G	H	I	J
Support of WikiPL functionalities	+	−	−	+	+	−	+	+	+

Table 1: Mapping of WikiPL functionalities to desired properties of SHARK tools

Table 1 provides an illustration of the arguments stated in the previous paragraph. The desired properties of SHARK tools are numbered as in §3.4, where also were defined. Property C is not present in the table because a mapping to wiki functionalities could not be established as discussed in §3.5. The second row of table 1 illustrates the support of WikiPL functionalities to the desired properties. When the plus symbol is present it means that WikiPL

functionalities do support the desired properties and when the minus symbol is present the WikiPL functionalities do not support desired properties.

6 Discussion

Our approach to use WikiPL as tool for SHARK can be further evolved to provide large-scaled architectural designs, though it could turn into a complicated task. Functions 1 to 3 used in §4.3 can be replaced by modules, or other nested designs. An architect can also design the architecture and just title it as an article, compose the documentation and set parameters and unit tests for the functions-to-be-developed.

Architects can commit their designs, while at the same time all participants can execute them, if applicable. Moreover the WikiPL provides semantic infrastructure, as it is a fork of MediaWiki. The current format of WikiPL does not use ontologies, but this semantic infrastructure could be used to set relations between design functions. Searching in the environment for certain architectural designs could facilitate the results of complex queries [11]. To conclude, our approach proposes designing article architectures that preserve the initial design via unit tests.

In the future there should be an implementation of WikiPL in an organization environment and a case study for the evaluation of the tool regarding SHARK should take place. Architects should be asked to use WikiPL to commit large-scaled designs by following our approach, which is mentioned in §4.3. There should be an examination on how other architects can use it as a collaboration tool and add comments, raise discussion and reuse the design-function articles. This would make the identification of extra needed functionality and drawbacks possible. Finally, there should be an examination on how useful the developers find the tool for deliverance of code, which follows the design guidelines by satisfying unit tests.

7 Conclusions

Wikis prove themselves efficient and effective as knowledge management tools as we defined in §3.2. It is a fact that wikis are the most popular content management tool at the time being, implementing several known applications for support in OS communities (Linux, FreeBSD), commercial communities (Nokia, IBM) and academic and learning communities; this can tell us a lot for their success. Our review findings verify the capability of wiki tools to support SHARK.

The approach of WikiPL supports software development knowledge. We examined the tool with respect to SHARK and simulated its support in small-scaled architectural designs. The results of this small simulation were positive but did not support well enough SHARK. Committing AK could become a complicated procedure, but on the other hand once it is composed in a right way it can be easily reused. The current implementation of the tool is also restricted in the Python programming language, which makes it more difficult be adopted by a large audience. On the other hand WikiPL functionalities support most of the desired properties of tools for SHARK, as discussed in §5.2.

Acknowledgements

The authors wish to thank Dan, Alexander and Alexandros.

References

[1] Wikipedia: The Free Encyclopedia, “Wiki”, 2011, http://en.wikipedia.org/wiki/Wiki [Online; Access date 16-03-2011]

[2] Cunningham, W., 2008, Wiki Design Principles, c2.com, http://www.c2.com/cgi/wiki?WikiDesignPrinciples [Online; Access date 16-03-2011]

[3] Clerc, V., de Vries, E., & Lago, P., "Using Wikis to Support Architectural Knowledge Management in Global Software Development", 5th Workshop on SHAring and Reusing architectural Knowledge, 32th International Conference on Software Engineering, ACM, 2010.

[4] Rennie, E., 2007, “Community Media in the Prosumer Era”, 3C Media Journal of Community, Citizen’s and Third Sector Media and Communication, Issue 3.

[5] Reinhold, S., “WikiTrails: Augmenting Wiki structure for Collaborative, Interdisciplinary Learning”, The 2006 Symposium on Wikis, pp. 47-57, Odense, Denmark, 2006. ACM, New York, NY, USA.

[6] De Boer, R. & Farenhorst, R., “In search of ‘Architectural knowledge’”. 3rd international workshop on Sharing and reusing architectural knowledge, New York, 2008.

[7] De Boer, R. & Farenhorst, R., “Architectural knowledge Management: Supporting Architects and Auditors”, 2009.

[8] Farenhorst, R., Lago, P., and Vliet, H.V., “Effective tool support for architectural knowledge sharing”, Software Architecture, 2007, pp.123-138.

[9] Henttonen, K. & Matinlassi, M. “Open source based tools for sharing and reuse of software architectural knowledge”, Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture, 2009, pp.41-50.

[10] Rus, I., Lindvall, M.: Knowledge Management in Software Engineering. IEEE Software vol:19, No:3, 2002.

[11] Shiva, Saijan G. & Shala, Lubna A. “Using semantic wikis to support software reuse”, Journal of Software [1796-217X] Shiva, 2008, vol:3, Issue 4.

[12] Schaffert, S., “IkeWiki: A Semantic Wiki for Collaborative Knowledge Management,” Proceedings of the 15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2006, pp.388-393.

[13] Wikipedia: The Free Encyclopedia, “About Wikipedia”, 2011, http://en.wikipedia.org/wiki/Wikipedia [Online; Access date 16-03-2011]

[14] Farenhorst, R., Lago, P., & Vliet, H.V., “ Experiences with a Wiki to Support Architectural Knowledge”, 3rd Workshop on Wikis for Software, 2008.

[15] WikiPL, “Welcome to WikiPL”, http://www.wikipl.com [Online; Access date 16-03-2011]

[16] Tang, A., Avgeriou, P., Jansen, A., Capilla, R., & Ali Babar, M. “A comparative study of architecture knowledge management tools”. Journal of Systems and Software, vol:83, No:3 2010, pp.352-370.

[17] Liang, P. & Avgeriou, P. “Tools and Technologies for Architecture Knowledge Management. In Software Architecture Knowledge Management”. Springer, Berlin, Heidelberg, 2009, pp.91–111.

[18] Freshmeat, “WikiPL”, http://freshmeat.net/projects/wikipl-10, [Online, Access date 16-04-2011]

[19] Farenhorst, R. & de Boer, R., “Knowledge Management in Software Architecture”, Software Architectural Knowledge Management: Theory and Practice, Springer, 2009, pp.21-38.

[20] Souzis, A., “Building a Semantic Wiki”, IEEE Intelligent Systems, 2005, vol:20, No:5, pp.87-91.

[21] Kousetti, C., Millard, D.E. & Howard, Y., “A Study of Ontology Convergence in a Semantic Wiki”, Proceedings of the WikiSym 2008, ACM, 2008.

[22] Hansen, M.T., Nohria, N. & Tierney, T., “What’s Your Strategy for Managing Knowledge?”, 1999, Harvard Business Review 77(2), pp.106-116.

[23] Kruchten, P., "The 4+1 View Model of Architecture", 1995, IEEE Software, vol:12, No:6, pp.42-50.

[24] Avgeriou, P & Zdun, U., “Architectural Patterns Revisited - A Pattern Language”, Proceedings of 10th European Conference on Pattern Languages of Programs (EuroPlop 2005), Irsee, Germany, pp.1-39, 2005.

[25] Avgeriou, P., Kruchten, P., Lago, P., Grisham, P. & Perry, D., “Architectural Knowledge and Rationale – Issues, Trends, Challenges”, ACM SIGSOFT Software Engineering Notes, 2007, vol:32, No:4, pp.41-46.

[26] R. K. Pandey, ”Architectural description languages (ADLs) vs UML: a review”. (UICSA) and R. D. University, 2010, Jabalpur (M.P.) India.

[27] van der Ven, J.S., Jansen, A., Nijhuis, J. & Bosch, J., “Design Decisions: The Bridge between Rationale and Architecture”, Rationale Management in Software Engineering, Springer, 2006, pp.329-346.

Variability Management in Business Processes: Comparison of approaches

Ntembeko Mkhunyana & Sinazo Matyila

Abstract — Business processes specify main activities in an organization, some of which can be automated. It is often the case that replication of activities across such processes occur and failure in identifying such issues result in organizational costs. This leads to huge modelling and maintenance efforts if they are not properly managed. In this paper, we discuss approaches used to manage variability in business processes and apply some of those methods into examples. We then compare these approaches and draw conclusions as to which method is better to implement in order to manage variants of business processes in organizations.

Index Terms — Variability Management, Business Processes, Provop approach, Process Variant Repositories (PVR), Compositional and Parametric approach.

1. INTRODUCTION

Business Process Management has been implemented in many organizations in order to maximize the quality and services provided to customers. The major reasons for implementing this was to:- (a) help an organization to get good profits in less time. (b) Visualize activities within the organization. (c) Visualize data flow within the organization. Without the use of BPM, organization would have not been able to achieve the points just mentioned above. Business Process Management is done with the help of using different tools that help in capturing, modeling, designing, integrating, deploying, testing, measuring, and managing several business activities. Each time a new requirement is introduced, that requirement has to be also included into the existing business process and the resulting process is a variant of the main process an organization initially had. The success or failure of an organization depends on how good or bad it is able to manage the entire life cycle of its business processes.

A business process is a collection of activities designed to produce a specific output for a particular customer or market [1]. Business processes play a major role in an organization as mentioned above because they specify the main activities that must be followed and they usually contain goals, specific inputs/ outputs, tasks that are performed in a certain order, and resources. The number of processes of a single organization can vary depending on how many departments an organization has and how many activities each department has to perform, and they can be very similar. It is often the case that replication of activities across such processes occur, however failure in identifying such issues results in organizational costs [3] as well as huge modeling and maintenance efforts that have to be done. So, we need a way to manage these processes.

Today there are methods and approaches that are widely used for handling and managing changes that take place in business processes and these will be discussed into detail in section 2 and 3 of this paper. Some of these approaches make use of the full Business Process Life Cycle and others just some parts of it. Based on [5], variability management is the set of activities aimed to cover the creation and support of differences in versions of reference processes. This variability in business processes can be managed using the following approaches: implementing the Provop approach [2], compositional and parametric approach with Aspect-Orientation [3], and through process variants repository (PVR) approach [4].

In this paper, we intend to present the methods or management strategies as mentioned above, in section 2. In section 3, we apply these methods to some of the examples given in [2], [3], and [5] which include production change process in the automotive industry, mobility & food allowance business process in the human resource domain, and Generic process for obtaining a subsidized wheel chair, respectively. Section 4, then compare these methods based on similarities with respect to advantages and dis- advantages. What we expect to find is the way to properly manage variability in business processes since they play a major role in every organization.

2. METHODS TO MANAGE VARIABILITY IN BUSINESS PROCESSES

This section describes methods used to manage variability in business processes. We make use of references [2], [3], [4] and [7] to discuss three approaches that can be used which are (a) The Provop Approach, (b) The Compositional and Parametric Approach, (c) Process Variants Repositories (PVR).

2.1 PROVOP APPROACH

Provop approach is an approach used to manage variability in business processes. It provides a very flexible solution to manage process variants in the full business process life cycle. The basic logic behind this approach is that, it takes all the variants in the process and capture or combine them to form one process model that consists of all those variant. For instance, if we let *P* represents a process model with variants *P1, P2 & P3;* and we let *T* represents tasks or activities, then we let *P1* to consist of tasks *T1a, T2a, T3a, T4a, & T5a; P2* with *T1a, T3a, T3b, T4a & T5a;* and *P3* with *T1a & T5a.* Syntactically, we can represent it like this:

a) *P: P1 → {T1a; T2a; T3a; T4a; T5a}*
b) *P: P2 → {T1a; T2a; fork: T3a, T3b; join: T5a}*
c) *P: P3 → {T1a; T5a}*

Then apply the Provop approach to variants *P1, P2, P3*, we can have a model consisting of process *P* with tasks similar to these: *T1a, T2a, T3a, T3b, T4a, T5a. that is: P → {T1a; T2a; fork: T3a, T3b; T4a; join: T5a}*. Provop approach is implemented using the full business process life cycle. In this section we discuss Provop approach for process variant management. The life cycle for modeling businesses consists of the following phases: Modeling phase, Instantiation and Selection, Deployment and Execution.

2.1.1 MODELING

Basic Process – The first step that Provop follow is to use the characteristics of process variants and look at their similarities to the original process model. This original process is usually called *basic process.*

Change Operations – In this stage, change operations are defined in the process. These change operations describe the differences between the basic process model and the respective variant model. The following are change operations that are applied to the variants: INSERT, DELETE & MOVE process fragments as well as MODIFY process element attribute.

Options – These are used to define more complex adjustments to processes and they are grouped into a single object called *options*.

This consist of a name and a set of change operations.

Visualization of options – This section visualize options as they are stored in the object named *options*. It shows all information of the option and enables user-defined selection of the information to be visualized.

Option Relation – This section describes relations that are applied after modeling different options. These relations are: *Dependency* means that the relation option is always dependent on the basic process; *Mutual exclusion* allows to reduce the possible combinations of options that can be applied to the basic process model; *Execution order constraint* allows specifying orders in which options can be applied to the basic process; & lastly, *Hierarchy* constitutes a combination of the relation *dependency* and *execution order*.

Context-aware Process Configuration - In a first step the process context has to be defined by utilizing context variables with a given range of value. Provop distinguishes between static and dynamic context variables. Static context variables are set once and their value is then fixed throughout process execution (for instance, product type). The value of dynamic context variables, in turn, may change during process execution (for instance, development phase).

Process Context Constraints - Sometimes there are constraints describing a relation between particular context variables. For example, if a requested product change is of high costs, its risks will be high as well. This follows the IF THEN ELSE logic.

Context Rules - To connect options with process variants configurations, a process context has to be defined. For this purpose, context rules are defined.

2.1.2 SELECTION AND INSTANTIATION

In this phase, three things are used, *that is*. *basic process,* defined *options* & the *context model* to configure models of different variants. The following are the steps taken to achieve this:

Step 1: Select Options – Here, relevant options are defined either explicitly or implicitly when configuring a process variant.

Step 2: Evaluate Relations – After selecting a set of options, their relation is checked. This checks if a dependent option is missing, to maintain consistencies in processes.

Step 3: Apply Options - After defining and evaluating the relevant set of options, the related change operations are applied to the model of the basic process. This process starts by applying static context variables, then dynamic context variables.

Step 4: Check Consistency – This step checks for redundancies and conflicts after applying changes in a basic process which may result in deadlocks later on.

2.1.3 DEPLOYMENT AND EXECUTION

In this section, the resulting variant model is translated into an executable work flow model.

2.2 COMPOSITIONAL AND PARAMETRIC APPROACH WITH ASPECT-ORIENTATION

In this section, the management of variability is based on a compositional and parametric approach based upon Aspect-Orientation. It leverages and extends an existing infra-structure with new transformations, modeling of relevant artifacts (business processes), their variability, and a new configuration knowledge mapping features expressions to such new transformations [3]. This approach is based upon the variability model of Modeling Scenario Variability as Crosscutting Mechanisms (MSVCM) which also provides a set of Haskell libraries [3]. This process works by configuring processes in such a way that after applying it the main process called *basic process* and *advice processes* are created. A *basic process* is the main process that is common to all the departments in that particular organization and an *Advice process* is the section in the process that was causing variability in the main process. This approach also present the BPMN extensions and transformations using the Haskell functional programming language [3].
Business Process Modeling Notation - is a graphical representation for specifying business processes in a business process model. An example of this functional programming language is given below:

```
data BusinessProcessModel =
  BPM { processes :: [BusinessProcess] }
  data BusinessProcess =
     BusinessProcess {
         pid :: Id,
         ptype :: ProcessType,
         objects :: [FlowObject],
         transitions :: [Transition
     }

  data ProcessType =
     BasicProcess |
     Advice {
        advType :: AdviceType,
        pc :: Pointcut
  }

  data FlowObject =
     FlowObject {
         fId :: Id,
         fType :: FlowObjectType,
         annotations :: [Annotation],
         parameters :: [Parameter]
  } | Start | End

  data FlowObjectType = Activity | Gateway
  data Pointcut = PC String
     type Transition = (FlowObject, FlowObject, Condition)
```

Listing 1: abstract syntax excerpt of BPMN extension in Haskell [3][7]

This syntax is a way of presenting an advice and a basic process after applying this approach. A clear example of this approach is presented in section 3.2 of this document. And the use of this syntax is shown in a small table in the results of the example.

2.3 PROCESS VARIANTS REPOSITORY (PVR)

A process variants repository (PVR) approach is the process of managing variants of a business process that are stored in the repository. This approach uses a concept of Business Process Constraint Network (BPCN) which reduces process specification to a set of minimal constraints . Detailed information about BPCN is not provided in this paper since the main focus is to present how PVR works in terms of managing process variants of a business process. All process variants of a business process satisfy the same set of constraints though they may be different. Process variants stored in a repository can result into a huge corporate resource which becomes valuable and provides knowledge to an organization. But the question how can we manage this process variant repository? The following will provide answer to this question.

2.3.1 MANAGING PROCESS REPOSITORIES

Once a process variant is stored into a repository, a *query* statement which is a statement of information needs, is formulated based on that variant. This statement will then be used later on when reducing these processes. A process repository creates a schema which defines the structure according to which process variants are stored. Based on R. Lu & S. Sadiq in [5], a process variant *V* is defined by process model *W*, where *W = (N, F)* defined through a directed graph consisting N: Finite set of nodes, F: Flow relation F is subset of NxN. Nodes are classified into tasks (*T*) and coordinators (*C*), where *C u*nion *T, C i*ntersection *T = e*mpty set. Task nodes represent atomic manual/ automated activities or sub-processes that must be performed to satisfy the underlying business process objectives. *C* allows the building of control flows structures, that is, fork, choice, loops, etc to manage the coordinator requirements. Let's consider the following

examples of variants stored in the process repository:

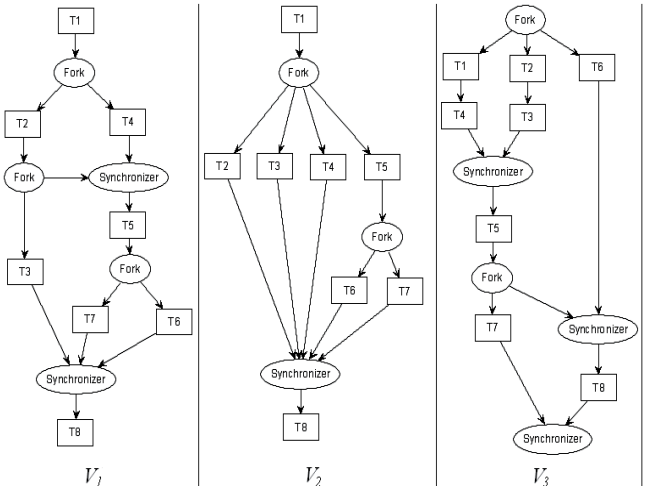


Figure 2.3-1: Examples of processes that show variability [4]

In the above processes, the constraint is as follows: in all process variants, *T1* must always be performed before *T5; T2* & *T4* must be done in parallel. Let's consider the following queries that can be applied to the variants shown above. Let the query can be defined as *Q*, then two query statements can be defined as *Q1* and *Q2*, respectively:

a) *Q: Q1 → {T1, fork, T2 & T3, join:Synchronizer, T8}*

b) *Q: Q2 → {T1, fork, T2, T3, & T4, join:Synchronizer}*

After defining the query statement, a test can be made to check if the the given variant is related with a specified query. An approach called SELECTIVE-REDUCE approach is used. This approach used graph reduction techniques to show the match. The steps that this approach follow are as follows, it:- (1) Eliminates all task nodes that are not contained in the query, (2) Reduces the flow relation using three reduction rules, namely *sequential* – eliminates all task nodes that are not part of the query statement*, adjacent* – redirects all adjacent "forks & synchronizes" controls into one control, and *closed* – if there are two flows coming from the same fork to the same synchronizer and one of them contains the node of interest, only the flow with the node of interest will be chosen. These set of rules are shown in [5], and examples for them are shown diagrammatically. After applying these rules to all process variants in Figure 2.3-1 (*V1, V2, V3)* for query *Q1*, gives a reduced structure *PV1, PV2 & PV3* as shown in Figure 2.3-3 below:

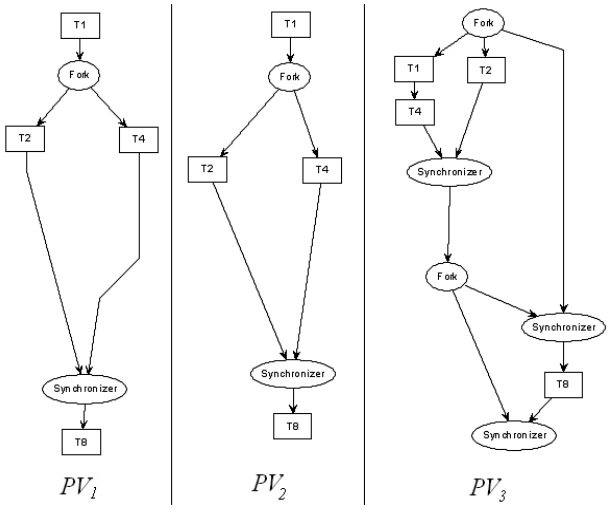


Figure 2.3-1: Outcomes of applying PVR to process variants [4]

3. IMPLEMENTATION OF METHODS

In this section we consider and discuss examples that clearly show variability in business processes and what bottlenecks it has in organizations if not managed well.

3.1 PRODUCTION CHANGE BUSINESS PROCESS

The first example that we look at is the production change business process from an automotive industry [2]. This business process is designed in such a way that it contains variants that are connected to each product type, for instance, a car, truck or a bus. All these business processes are designed with a specific goal in mind that has to be achieved. In this example, let's consider the following business processes as shown by figure 3-1 below, starting from 1a to 1d, for the production change process.

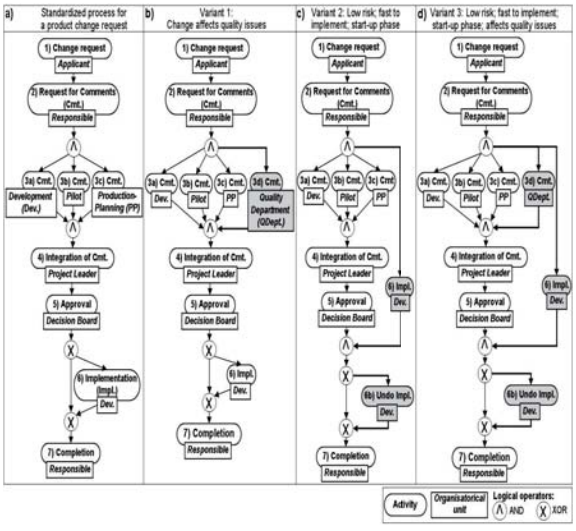


Figure 3-1 Production Change Process [2]

Considering business process in (a), there is a change that is about to take effect in a certain domain as shown by *Activity 1: Change request*. The person responsible for coordinating changes request comments from all the departments that are about to be affected by this change. These departments are shown by *Activities 3a, 3b, 3c* and the request for comments is issued by *Activity 2*. Once the comments have been collected, an integration change document is created by the project leader as shown in *Activity 4*, then it's passed

on to the decision board for approval, *Activity 5*. Once the document has been approved by the decision board, implementation process takes place, *Activity 6*; or else this step is skipped and the change request gets filled. The process ends. Looking at business process in (b), there is an additional department involved i.e. the quality department, *Activity 3d*, for considering quality critical issues for the requested change. Comments from this department play an important role. Business process in Figure (c) shows a fastened process of implementing the requested change. *Activity 6* in Figure 1b is now executed before the approval of the decision board. If the board refuses to approve the change request, *Activity 6b* is executed and the implemented change has to be undone. Business process in Figure (d), will be required if the process affects quality critical issues but still needs to be fastened. This business process is the combination of Figure 1b and 1c and it inherits all the adjustments and activities from these two processes or variants.

When processes have variants like the ones shown in Figure 1a to 1d, they are usually kept in a separate process model as in Figure 1. This results in huge amount of redundancies as variants are identical for most parts, and there is no support for automatically combining models into one. This makes it difficult for process designers to analyze and combine these processes. To provide the solution to this problem, we apply a method that has been discussed before called the Provop (PROcess Variants by OPtion) approach for managing large collections of process variants and make one model out of them. The first step is to define figure 3-1 as the basic processes, then look for alternatives that could join all variants into one mode and apply Provop phases as discussed in section 2.1 of this document to Figure 3-1. The result will be a single model consisting of all the variants as shown below:

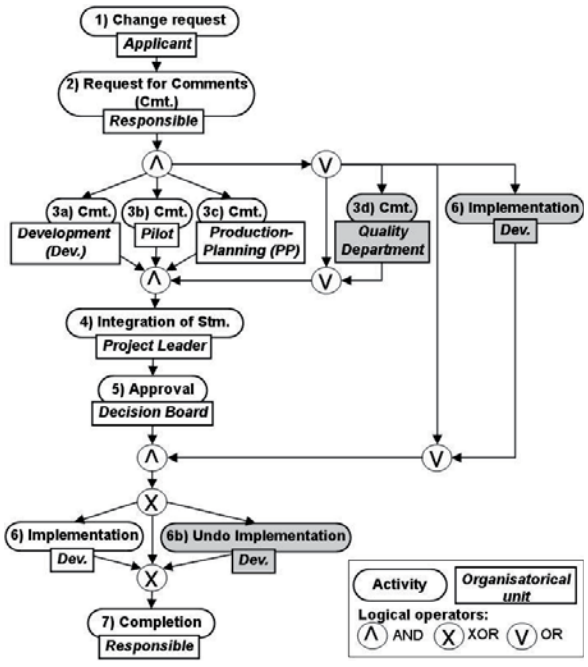


Figure 3-2: The result of the business process using Provop approach [2]

3.2 MOBILITY AND FOOD BUSINESS PROCESS

In this example, we show how to manage variability of business processes in the Human Resource domain. The variability analysis focused on identifying variability patterns involving activities within processes sharing a significant amount of similarity. A total of approximately sixteen fine-grained variability patterns were identified, which could be further classified into the following coarse grained patterns: 1) insert/ removal/ replacement of activity/ flow of activities before/ after/ around activity/ gateway/ sub-process; 2) parameter value variability within flow objects; 3) variability of lanes to which activities belong [4]. The following figure (see below, Figure 1) illustrates the model that resulted in the analysis made by analysts in a human resource domain. Two business processes that were analyzed in this domain are shown below, that is, the Food Allowance and Mobility Allowance. The results are shown below in Figure 3-3 and Figure 3-4:

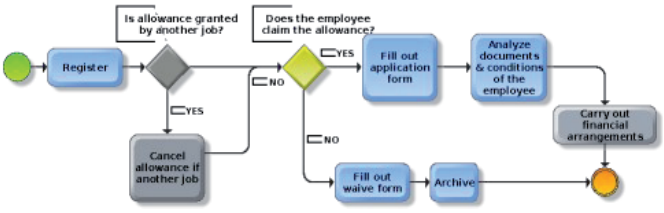


Figure 3-3: Food Allowance business process [3].

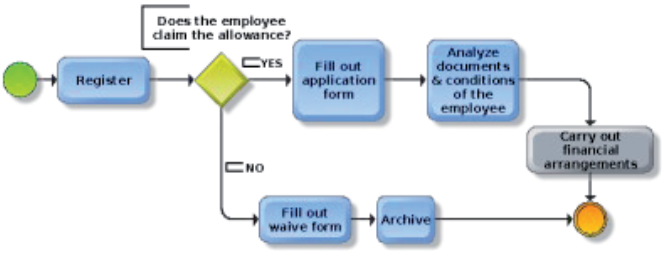


Figure 3-4: Mobility Allowance business process [3].

In the above figures, that is 3-3 and 3-4, variability is shown by gray areas. In figure 3-3, there is an added activity right after registration and this makes the food allowance process differ from mobility allowance process, though other activities are exactly the same. Such processes represents variability in Human Resources domain business processes. Taking from the discussion in section 2.2 above, that is *Compositional and Parametric approach with aspect-orientation*, we apply this method in this example.

The first step is to separate the common and variant behavior in business processes in figure 3-3 and 3-4, that is, the method breaks the business process into two parts, a *basic* business process with a shared objects and transitions, same as Figure 3-4, and an *advice* formed by the variant assets. This result in the following processes is the same Figure in Figure 3-4 together with the variant shown below:

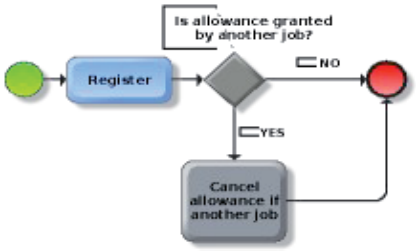


Figure 3-5: *Advice configuration process* – variant amongst Allowance business processes and fragmentation of the configuration knowledge (on the right side, respectively) [3].

The table below also forms a result of this approach, along with Figure 3-5

Feature Expression	Transformations
Allowance	**selectBusinessProcess** bpCommonAllowance
Food Allowance	**evaluateAdvice** advFoodAllowance

Table 1: Fragment of the configuration knowledge [3]

In this approach, all advice processes are kept aside from the basic process. This help in order to have 1 common business process, then variants belonging to that process separately, as advice processes. The next step is to enable the configuration of the common business process. We have to relate the feature expression *Allowance* to the transformation *selectBusinessProcess "bpCommonAllowance"*, where *bpCommonAllowance* is the identifier of the business processes that handles the commonality among the allowance processes. Again, to evaluate the Food Allowance advice (Table 1), we have to relate the feature expression *Food Allowance* to the transformation *evaluateAdvice "advFoodAllowance"*, where *advFoodAllowance* is the identifier of the *Food Allowance advice* declared in the SPL assets. Table 1 shows a fragment of the configuration knowledge with these transformations.

3.3 PROCESS FOR OBTAINING A SUBSIDIZED WHEEL CHAIR

In this example, we consider the Netherlands as having 441 municipalities that has to implement the same national law but these municipal sites have different sizes, IT infrastructure, business model, etc. In 2007, the Wet maatschappelijke ondersteuning, Social Support Act was approved to provide public subsidized wheel chair to people in need. There were two ways of managing this national law, the first one was to let each municipality implement the lay using the interpretation document. The second was to make a formal and generic process then send it to municipalities so that they can customize it according to their organizational and IT structure. Figure 3-6 shows the second way of implementing this law.

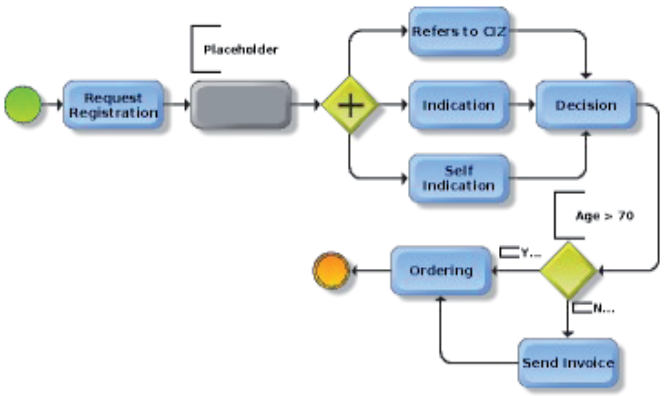


Figure 3-6: Generic process for obtaining a subsidized wheel chair. [5]

In this business process, we see an activity for registering for a subsidized wheel chair, then there is a placeholder in activity 2. After that, there is a decision to be made by an authorized civil servant. Based on the decision made, one of the three options have to be chosen which are to determine the need for the wheel chair. Then the decision to grant based on age is made. If the age is > 70, then the wheel chair will be ordered on subsidy, else an invoice will be made for that citizen and pay for the wheel chair.
Then, the following process (see below: figure 3-7) is the variant of the process in figure 3-6. This is based on the wishes of other municipalities when they customize the process. A home visit by the municipal authority will be done after registration to assess the situation before the wheel chair can be bought. In the Netherlands it is possible for a municipality to outsource the indication to an organization known as the CIZ (Centrum Indicatiestelling Zorg) for handling the indication. Then, the last change to be made was to change the age requirements from 70 to 65.

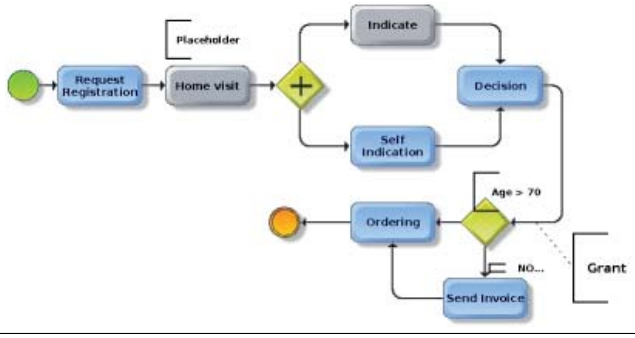


Figure 3-7: Process variant for obtaining a subsidized wheel chair. [5]

In solving this variability problem of these business processes, we apply PVR to this example as our third approach to manage the variability. This process reduces a process into a more manageable and understandable process showing only the activities that cannot be skipped. For instance, a situation has to be assesses in order to grant a wheel chair, and two activities, that is, indication and self indication are common in both processes but only one has to be chosen at a time. The result of implementing a PVR approach leads to the following:

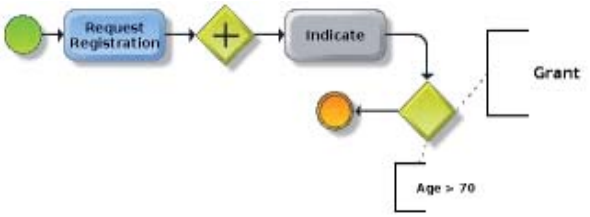


Figure 3-8: Results of applying PVR to the variant in Figure 3-6 [5]

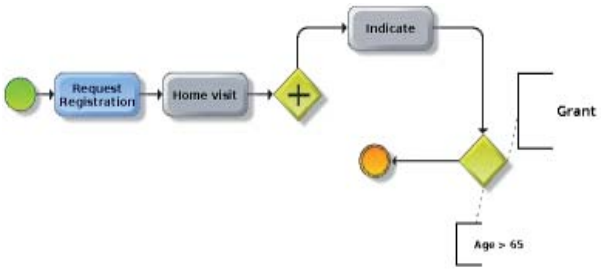


Figure 3-9: Results of applying PVR to the variant in Figure 3-7 [5]

4. EVALUATION

Advantages of Provop Approach: Information is provided in a more unified way to users since processes variants become a single process. Maintains consistencies. Easier to learn and implement. It follows exactly the full steps of a business process life cycle which makes it easier to understand.
Disadvantage of Provop Approach: At times it can become too complex and hard to analyze if it is implemented in a large organization that has many departments executing their own processes.

Advantages of PVR: It stores the process in a query for all the processes that are running, and it keeps information in options which makes it easy to communicate it to other stakeholders. It is easy to understand since it reduces processes and only the relevant activities remains which are those that are in the constraint. Process repositories store high volume of processes which becomes useful and valuable as a corporate resource. The information is modeled in a structured format.
Disadvantage of PVR: Redundancies may occur in the repository if not documented and managed properly.

Advantages of Compositional & Parametric approach with aspect orientation: It simplifies processes by detaching the section that causes variability in the process and leaves the main process which is common to all departments in an organization.

Disadvantage of Compositional & Parametric approach with aspect orientation: If the sections that are seperated from the main process are not documented properly, they can cause lots of confusion to other stakeholders.

5. CONCLUSION

In this paper, we discussed three approaches namely, Provop approach, PVR, and Compositional and Parametric approach. Then, we showed how these approaches are used to solve variability in business processes using examples taken from different sources. After applying the approaches mentioned above to the examples, we compared them based on their advantages and disadvantages in order to determine which approach is better and simple to use. The results we got after doing an analysis is that not all of these approaches are suitable to all organizations. For instance, Provop approach combines all process variants into one model. So, using this approach in a large organization can result in huge process analysis problems since the final process would be a big process spanning all the variants. Hence, this process can be best used in small organization. On the other hand, PVR only focuses on the main important activities of the organization, hence, eliminating some of the activities in the process model. This one is works fine for both small and large organizations. The last approach, compositional and parametric approach is most suitable for large organizations with lots of processes that are being implemented but it also fits well in solving variability issues in small organizations business processes.
In comparing these approaches, all of these approaches are suitable for certain organizations depending of the size and the number activities the business performs. *Further work on this issue:* We are still intending to further study in these processes and find out about other approaches that are far more better than these one discussed here. Then we can again produce more results based on those processes.

ACKNOWLEDGEMENTS

We would like to thank all the anonymous reviewers for reviewing our paper and providing feedback so that we can improve our work.

REFERENCES

[1] Sparx Systems, *"The business Process Model",* www.sparxsystems.com.au. [online], 2004
[2] A. Hallerbach, T. Bauer, and M. Reichert, *"Managing Process Variants in the Process Life Cycle*", in Proceedings ICEIS (3-2), 2008, pp.154-161.
[3] I. Machado, R. Bonifácio, V. Alves, L. Turnes, and G. Machado. *"Managing Variability in Business Processes: An Aspect-Oriented Approach"*, in Proceedings EA '11, 2011, pp.25-30.
[4] R. Lu and S. Sadiq. *"Managing Process Variants as an Information Resource"*, in Proceedings Of 4th Int. Conf. On BPM, 2006, pp.426-431.
[5] M. Aiello, P. Bulanov, H. Groefsema. *"Requirements and Tools for Variability Management"*, in Proceedings COMPSACW 2010 IEEE 34th Annual Conf, 2010, pp.245-250
[6] Lu, R., Sadiq, S., Padmanabhan, V., Governatori, G.: *"Using a Temporal Constraint Network for Business Process Execution"*. In: Proceedings 17Th, Australasian Database Conference (ADC2006), 2006, pp.157-166.
[7] K. Czarnecki and U. Eisenecker. *"Generative Programming: Methods, Tools, and Applications"*. Addison-Wesley Professional, 2000.

Comparison of MapReduce implementations

René Zuidhof & Jos van der Til

Abstract—Processing large data sets can be very time comsuming, especailly when only using one computer. To decrease this processing time, one can try to share the processing time over multiple computers in a cluster. Therefore a framework is introduced for splitting (mapping) and gathering (reducing) large tasks so that they can be parallelly processed in an distributed environment. Since the introduction of the framework multiple implementations have been developed, each with their own features and limitations. In this paper we compare three of the MapReduce implementations for some important features like performance, applicability and scalability. The results are these features with their own 'winning' implementation.

Index Terms—MapReduce, Hadoop, CouchDB, MongoDB, Distributed Computing, Parallel Computing

1 Introduction

Although computers are getting better and faster, processing large data sets on one computer can be very time consuming. A solution to this problem is to use multiple computers to solve one problem. This is why the MapReduce framework was introduced. Some examples of the MapReduce implementations are Hadoop, CouchDB and MongoDB. In this paper we describe and compare the features and limitations of these implementations. these comparisons are based on the information available on the internet aswell as on experiments done by ourselves.

1.1 MapReduce

MapReduce is a software framework introduced by Google to support distributed computing on large data sets using clusters of computers. Using this framework large quantities of data can be processed in a short amount of time by doing two steps. The first step is the mapping step, here the main task is splitted into multiple subtasks. These subtasks can then be processed on the computers in the cluster and will all result their own partial answer. The second step is to gather and combine the processed subtasks to produce the final solution, this is the reduce step. These steps can be compared to the divide & conquer steps used in other algorithms, for example MergeSort, and is therefore not entirely new. But using this on a larger scale and over clusters of computers, a new dimension of data processing is reached with its own problems and difficulties. The MapReduce framework is used by Google to completely regenerate Googles index of the World Wide Web, and replaced the old ad hoc programs that updated the index and ran the various analyses. Other uses of the framework are distributed grep, distributed sort, web link-graph reversal, web access log stats, document clustering and inverted index generation. [14] This last example will be discussed in section **??** where we will analyse features like performance, applicability and scalability for the different implementations of the MapReduce framework.

1.2 Map

A typical MapReduce framework takes key/value pairs as arguments for their Map function. Using these pairs it will generate zero or more output pairs. For our inverted index experiment the input for a Map function would be a ¡line, docid¿ pair which would result in an output with multiple ¡word, docid¿ pairs.

- *René Zuidhof, Computing Science student (Software Engineering & Distributed Systems) at the University of Groningen, e-mail: h.j.zuidhof.1@student.rug.nl.*
- *Jos van der Til, Computing Science student (Software Engineering & Distributed Systems) at the University of Groningen, e-mail: j.r.van.der.til@student.rug.nl*

1.3 Reduce

These output pairs are used to gather the input for the Reduce function. This function is called for each unique key. The Reduce function than gathers all values associated with the given key and returns these as the ouput. For the inverted index the result would be a word with all the document id's in which the word was present.

2 Apache Hadoop MapReduce

Apache Hadoop MapReduce is a programming model and software framework for writing applications that rapidly process vast amounts of data in parallel on large clusters of compute nodes. [9] It is part of the Apache Hadoop project, which aims at providing open-source software for reliable, scalable, distributed computing. [4] The Hadoop framework was inspired by Google's MapReduce and Google File System papers. The project is supported by a global community of contributors of which Yahoo! has been the largest contributor to the project and still uses Hadoop for their services. Hadoop was created by Doug Cutting, who named it after his son's stuffed elephant. [18]

2.1 Architecture

The core of the Hadoop framework is Hadoop Common, a set of utilities that support the Hadoop subprojects. This includes filesystem utilities, Remote Procedure Call (RPC) and serialization libraries. The filesystem utilities and libraries in Hadoop Common merely provide access to the supported filesystems and should not be confused with Hadoop Distributed File System (HDFS).

For effective scheduling of work all filesystems used by Hadoop should provide their location. By doing so, they allow the server to schedule jobs on servers closest to the data and thus reducing bandwidth usage. One of the filesystems that supports location awareness is the HDFS, more on this in section 2.2. [11]

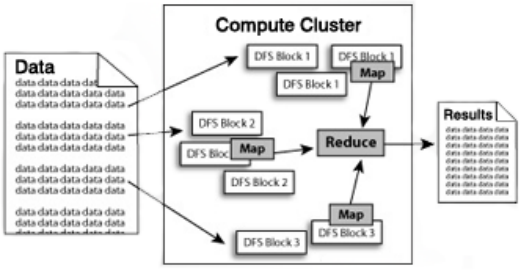


Fig. 1. Example of an Hadoop infrastructure [7]

A typical Hadoop cluster will include a single master and multiple slave nodes as shown in figure 1. The master node consists of a jobtracker, tasktracker, namenode and datanode. A slave or compute node consists of a datanode and tasktracker, as can be seen in figure

2. Hadoop requires JRE 1.6 or higher and SSH to be set up between nodes in the cluster. [18, 11]

2.2 Hadoop Distributed File System

The HDFS is a distributed, scalable, and portable filesystem written in Java for the Hadoop framework. Each node in an Hadoop cluster typically has a single datanode, this situation is typical because a node does not require a datanode to be present. Using the TCP/IP layer for communication, the HDFS specifies a block protocol to send data between the cluster of nodes. Using a file size of, ideally, a multiple of 64MB, large files are stored multiple times and over multiple nodes, on default files will be replicated three times. Although this might sound redundant this is done to achieve reliablity. Data nodes can talk to each other to rebalance data, to move copies around, and to keep the replication of data high.

The filesystem requires one unique server, the NameNode. This is a weak point in the filesystem because if this node goes down the entire filesystem is down. When the node comes back up it must replay all outstanding operations which can take up over half an hour. To reduce the time after a shutdown a Secondary NameNode is included, not to take over when the Primary NameNode goes down, but to help the Primary NameNode after a restart. This is done by taking snapshots of the Primary NameNode's directory information, which is then saved to local/remote directories. When the Primary NameNode restarts it can use this directory information to restart without replaying all outstanding operations.

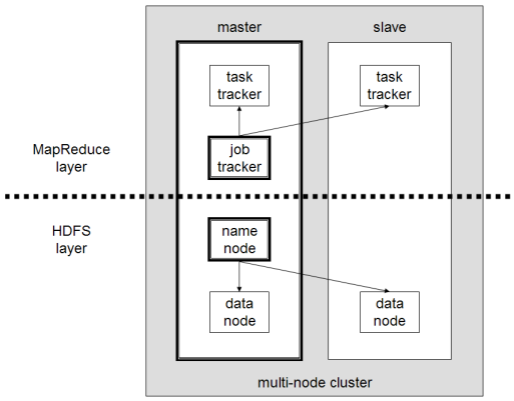


Fig. 2. A multi-node Hadoop cluster [18]

A key feature of the HDFS is data awareness between jobtracker and tasktracker. The jobtracker schedules map/reduce jobs to tasktrackers with an awareness of the data location. An example of this would be if node A contained data (x,y,z) and node B contained data (a,b,c). The jobtracker will schedule node B to perform map/reduce tasks on (a,b,c) and node A would be scheduled to perform map/reduce tasks on (x,y,z). This reduces the amount of traffic that goes over the network and prevents unnecessary data transfer. [18]

Hadoop can be used without the HDFS by using an alternative filesystem. Examples of such filesystems are the FTP filesystem or the HTTP(S) filesystem. When one of these filesystems are used the feature of data awareness is not avaliable. This can have a significant negative impact on the performance of job completion times, which has been demonstrated when running data intensive jobs. [19]

Another limitation of HDFS is that it cannot be directly mounted by an existing operating system. Getting data into and out of the HDFS file system, an action that often needs to be performed before and after executing a job, can be inconvenient. [8]

File access can be achieved through the Java API, the Thrift API, the command line interface, or browsed through the HDFS User Interface webapp over HTTP. Thrift is a software framework for scalable cross-language services development. It combines a powerful software stack with a code generation engine to build services that work efficiently and seamlessly between C++, Java, Python, PHP, and Ruby. [6]

2.3 Features and limitations

This section covers some features and limitations.

Features

- Data awareness; Hadoop in combination with HDFS knows which computer is closest to the data and will use this information to schedule the work in a way that the data is close to the actual process.
- API for multiple programming languages.
- Reliablity by multiplying files and save them over multiple nodes.

Limitations

- If the server NameNode goes down the filesystem is offline
- For maximum parallelism, you need the Maps and Reduces to be stateless, to not depend on any data generated in the same MapReduce job. You cannot control the order in which the maps run, or the reductions.
- It is very inefficient if you are repeating similar searches again and again. A database with an index will always be faster than running an MapReduce job over unindexed data. However, if that index needs to be regenerated whenever data is added, and data is being added continually, MR jobs may have an edge. This inefficiency can be measured in both CPU time and power consumed.
- In the Hadoop implementation Reduce operations do not take place until all the Maps are complete (or have failed and been skipped). As a result, you do not get any data back until the entire mapping has finished.

3 CouchDB

Apache CouchDB is a document-oriented database that can be queried and indexed in a MapReduce fashion using JavaScript. Written in Erlang, a robust functional programming language ideal for building concurrent distributed systems. Erlang allows for a flexible design that is easily scalable and readily extensible. CouchDB provides a RESTful JSON API than can be accessed from any environment that allows HTTP requests. Third-party client libraries are available that make it easier to use CouchDB for different programming languages. CouchDBs built in Web administration console speaks directly to the database using HTTP requests issued from your browser. [11]

A CouchDB document is an object that consists of named fields. Field values may be strings, numbers, dates, or ordered lists and associative maps. An example of a document would be a forum post:

```
"Subject": "CouchDB and MapReduce"
"Author": "John"
"PostedDate": "20/3/2011"
"Tags": ["MapReduce", "CouchDB"]
"Body": "How do I use MapReduce functions
  in CouchDB"
```

A CouchDB database is a flat collection of these documents. Each document is identified by a unique ID. Unlike SQL databases which are designed to store and report on highly structured data, CouchDB is designed to store and report on large amounts of semi-structured, document oriented data. A built-in conflict management system is provided and the replication process is efficient and fast, copying only documents and individual fields changed since the previous replication. With CouchDB, no schema is enforced, so new document types with new meaning can be safely added alongside the old. The view engine is designed to easily handle new document types and disparate, but similar, documents.

3.1 Views

Views are the primary tool used for querying and reporting on CouchDB documents. They are built dynamically using JavaScript and do not affect the underlying document. This means that the views are not built when a file is saved but rather when the file is accessed. This can cause the first access to take some time depending on the amount of data. [13]

3.2 MapReduce

In CouchDB, each view is constructed by a JavaScript function that acts as the Map half of a MapReduce operation. The Map function transforms each document into zero, one or multiple intermediate objects, where the Reduce function is another user defined function to combine the intermediate objects into the final result. The intermediate objects of the Map and the Reduce function are stored in the view indexes. As the storage gets updated, the previous results stored in the view indexes is no longer valid and has to be updated. [11]
Figure 3 shows the workflow of CouchDB with an explanation below.

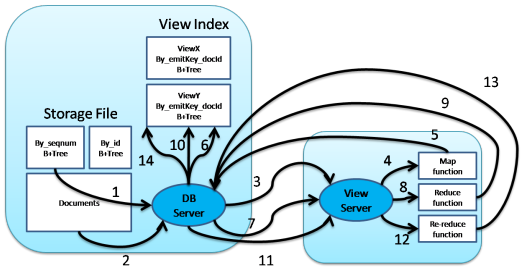


Fig. 3. The CouchDB workflow [10]

1. CouchDB will walk the by_seqnum B+Tree index of the storage file. Where B+Tree is a type of tree which represents sorted data in a way that allows for efficient insertion, retrieval and removal of records, each of which is identified by a key.
2. Based on that, CouchDB get the latest revisions of all existing documents.
3. CouchDB remembers the last seqnum and then feed each document to the View Server using map_doc.
4. View Server invokes the map(doc) function, for each emit(key, value) call, an entry is created.
5. Finally, a set of entries is computed and returned back to CouchDB.
6. CouchDb will add those entries into the B+Tree index, key = emit_key + doc_id. For each of the B+Tree leave node.
7. CouchDB will send all its containing map entry back to the View Server using reduce.
8. View Server invokes the reduce(keys, values) function.
9. The reduce result is computed and returned back to CouchDB.
10. CouchDB will update the leave B+Tree node to point to the reduce value of its containing map results.
11. After that, CouchDb moves up one level to the parent of the leave B+Tree node. For each of the B+Tree parent node, CouchDB will send the corresponding reduce result of its children nodes to the View Server using rereduce.
12. View Server invokes the reduce(keys, values) function again.
13. Finally a rereduce result is computed and returned back to CouchDB.
14. CouchDB will update the parent B+Tree node to point to the rereduce value.

The rereduce is needed to reduce multiple intermediate results into one. [10]

3.3 Scaling

The standard CouchDB implementations does not cover the possibility to work in a cluster. Therefore CouchDB Lounge is needed. Lounge is a proxy-based partitioning/clustering framework for CouchDB. It provides redundant storage by duplicating data over atleast two nodes to guarantee reliability. Besides the duplicating of information Lounge also takes care of splitting partitions when they get to big. [12]

3.4 Features and limitations

Features

- API for most of the popular programming languages.
- Distributed Architecture with Replication: CouchDB was designed with bi-direction replication (or synchronization) and offline operation in mind. That means multiple replicas can have their own copies of the same data, modify it, and then sync those changes at a later time. The biggest gotcha typically associated with this level of flexibility is conflicts.
- ACID Semantics: Like many relational database engines, CouchDB provides ACID semantics. It does this by implementing a form of Multi-Version Concurrency Control (MVCC). That means CouchDB can handle a high volume of concurrent readers and writers without conflict.
- Fast atomic updates on documents (concurrent modifications of single documents).

Limitations

- No standard scaling possibility. To get CouchDB to work in a cluster Lounge has to be installed.
- No data awareness
- Just like the standard scaling possibility. It needs more third party software to get some other features.

4 MongoDB

MongoDB is quite similar to CouchDB, it is also a collection-oriented, schema-free document database. Like CouchDB documents consist out of key-value pairs, however the serialization of the data is different. Where CouchDB uses JSON for serialization of data, MongoDB uses BSON or 'Binary Serialized dOcument Notation'. Like Hadoop and CouchDB, MongoDB can be run in a cluster of independant MongoDB 'shards'. An architectural overview of such a setup is visible in figure 4, showing multiple shards, configuration servers and routing processes. Since the sharding model is order preserving, adjacent, as defined by shard key, tends to be on the same server. Thus reducing bandwidth usage and response time when the data is processed. However, this does require that a good shard key has to be chosen, ensuring that data is distributed evenly across the multiple shards.

In MongoDB MapReduce functions are written in Javascript, this is because all MapReduce tasks run inside a Javascript engine. The parameters to start a MapReduce task include: the collection that stores the data to process, a map function, a reduce function and a destination collection to store the results. Map functions feed data to the reduce functions by performing an `emit(key, value)` call. Reduce functions are of the following form: `reduce(key, value_array)`, receiving a key and a list of values. This operation should

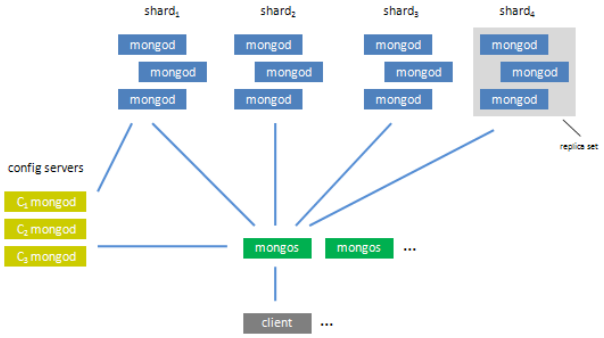


Fig. 4. MongoDB sharding architecture [16]

be idempotent, e.g.: `reduce(key, value_array) = reduce(key, reduce(key, value_array))`. MapReduce tasks on large collections can benefit greatly from a sharded architecture, as MapReduce tasks are run in parallel across all shards. This is especially necessary for large data operations since MapReduce tasks on a single `mongod` process are single threaded, this is due to a design limitation of current Javascript engines. [15]

4.1 Features and limitations

Features

- Tuneable sharding architecture.
- Replication and High availability support.
- Enterprise support available.
- Fast atomic updates on documents

Limitations

- Authentication methods are extremely basic at best, no authentication when using replication or sharding.
- No support for storing data in a compressed format
- Document size limit of 4 MB in earlier versions, and 16 MB in newer versions.

4.2 Comparison

This section covers the comparison of the different features and limitations of the different implementations. The comparison is based on a set of features, these features are performance, applicability, scalability and reliability.

4.3 Comparison of features

Because we compare the implementations for their MapReduce performance we only discuss the methods which have effect on these processes. On default we will discuss the standard implementations if not stated otherwise. This is done because Hadoop and the other two implementations are not comparable in the way they do their work, e.g. a software framework against a document-oriented database.

Features	Hadoop	CouchDB	MongoDB
Distributed MapReduce	yes	no[7]	yes
Data awareness	yes[1]	no[2]	yes
API's for most popular programming languages	yes	yes	yes
Indexed data	no	yes	yes
Data replication	yes	yes	yes
Sharding/Splitting data	yes	no[6]	yes
Enterprise support	no[3]	no	yes
Authentication	no[5]	yes	yes[4]

1. *With its standard file system, HDFS*
2. *Possible when using GeoCouch.*
3. *Third party, but only on their distribution.*
4. *Not when using sharding or replication, trusted security environment is recommended.*
5. *Not yet, future releases will.* [3]
6. *yes when using CouchDB Lounge.*
7. *yes when using BigCouch.*

4.4 Performance

Because of the file system used by Hadoop (HDFS), data awareness is supported. This is one advantage when checking the performance. Because of this feature the processing will be done by the nodes who hold the data, minimizing the time needed to transport data from node to node. One minor thing which can be discussed for Hadoop is the programming language. Hadoop is written in Java, compared to the other programming languages used in the other implementations (erlang and c++), Java is the slowest. An advantage of CouchDB compared to Hadoop is that it is a document-oriented database. This means that the data is indexed which gives a good advantage for the performance. An disadvantage for CouchDB is the lack of data awareness. This means a lot of time might be spend to transport data from one node to another. MongoDB has neither of these disadvantages. It is a document-oriented database, written in c++ and does support data awareness.

4.5 Applicability

This section covers the applicability of the implementations. This is based on things we ran into during our own experiment aswell as the documentation found on the internet.

Hadoop is specially developed for MapReduce processing. Setting up a single node process is not a hard thing to do (on linux). But when going from one node to multiple nodes, the configuration gets a lot harder. There are a lot of config files (XML) which need to be set. There is no support for authentication, yet. Apache Hadoop announced that this will be added in future versions [3]. API's for most popular programming languages are supported. Good documentation is available.

Unlike the other two implementations, CouchDB does not support distributed MapReduce in its standard implementation. Besides the lack of the standard distributed MapReduce features, it also does not support data awareness and the sharding of data. For these features third party software is required, like BigCouch and Lounge. Setting up a CouchDB node is not a lot of work if all its prerequisites are already installed. Installing these can be a lot of work because some of them have to be build from source. CouchDB is the only of the three implementation which supports authentication by default. Like Hadoop, CouchDB also provides different API's for most popular programming languages.

Unlike CouchDB, MongoDB supports most features needed for distributed MapReduce processing by default. Authentication is available, but can only be used when the sharding and replicating features are not used. When using on of these features, MongoDB recommends a trusted cluster. Just like the other two implementations, API's are available for most popular programming languages.

4.6 Scalability

This subsection will cover the scalability features for the various implementations. Since MapReduce is intended to be used on large clusters of computers, having a solution that is easily scaled across multiple nodes is preferred.

Hadoop provides a clustering solution by default, however it should be combined with the usage of HDFS to attain the best performance. This is due to the location and data awareness features that HDFS offers. Typically a Hadoop cluster contains one machine as the NameNode and one machine as the JobTracker (primary nodes), the rest of the nodes are DataNodes *and* TaskTrackers (slave nodes). [2] The JobTracker is the first point of failure in the Hadoop cluster, if it goes down all running jobs are halted. The NameNode is another point of failure inside the cluster, if this goes down the entire HDFS filesystem will go offline, and thus no data is available for the MapReduce tasks. The NameNode can be supported by a SecondaryNameNode, but this offers no real redundancy since it only creates checkpoints of the file system. A BackupNameNode is planned for Hadoop 0.21+, but there are currently not enough active contributors to make it Highly Available. [5]

Since MongoDB is a document database, the clustering solution involves splitting the data over multiple nodes, this is a process called sharding. This is because each node contains a 'shard' of the entire database, in this context we will refer to a node as a shard.

- Automatic balancing for changes in load and data distribution
- Easy addition of new machines
- Scaling out to one thousand nodes
- No single points of failure
- Automatic failover

Unlike Hadoop or MongoDB, CouchDB doesn't provide a scaling mechanism by default. However, third-party sharding mechanisms for CouchDB are provided. One example is CouchDB Lounge, which acts as a proxy (much like the mongos process from MongoDB) for incoming requests. Another example would be BigCouch, which looks like a fork of the original CouchDB code, adding clustering and sharding functionality to CouchDB.

4.7 Reliability

Reliability is measured in terms of data protection, this is because all three MapReduce implementations are able to store large quantities of data. A solution that provides a good scheme for protecting data against corruption is preferred.

Hadoop provides data integrity through its HDFS file system, by using replication. It should be noted that a replication factor of three or larger should be chosen, this is because silent corruption can occur within the system. If corrupt blocks are not detected in a timely manner, or if a software bug masks invalid blocks, then the system thinks there are more valid blocks than there actually are. With only one backup replica (replication factor of two), it is only when the backup fails that the system detects that there are no block replicas, at which point it is too late to recover data. With three replicas the system has a chance to detect the silent corruption after the second replica fails, and to re-replicate using the third replica. [17]

The greatest weakpoint of the Hadoop cluster is the NameNode, if it fails, then the whole HDFS cluster is unusable. If the NameNode proves unrecoverable, then all of the data in the cluster is unrecoverable. In a data critical context this is catastrophic, a intricate backup and recovery plan for the NameNode metadata should be in place. A secondary NameNode is recommended and, should run on a separate node to the primary. In the case of losing all of the primary's data (local disks and backups), the secondary can provide a stale copy of the metadata. Since it is stale, there will be some data loss, but it will be a known amount of data loss, since the secondary makes periodic backups of the metadata on a configurable schedule.

MongoDB provides a reliability mechanism in two ways. The first is by writing all actions to a journal (journalling) before performing them, and removing the journal once all the data is committed. The second way is by using replication to replicate data over multiple servers within a shard. Since each shard is actually a replication set of multiple servers, and thus each server in a replication set contains all the data stored in the shard. Sharding also uses automatic failover to deliver high availability. MongoDB currently has no way of knowing if silent data corruption occurs eg. by drive corruption.

MongoDB metadata is not as vulnerable as with Hadoop, since configuration servers (the servers that store the metadata) can be run in a automated failover cluster. Each config server has a complete copy of all chunk information. A two-phase commit is used to ensure the consistency of the configuration data among the config servers. Note that config server use their own replication model; they are not run in as a replica set. If any of the config servers is down, the cluster's meta-data goes read only. However, even in such a failure state, the MongoDB cluster can still be read from and written to. [1]

The replication features of CouchDB are used in two ways, scaling horizontally and replication of data for data security. Concurrency of CouchDB is very strong due to it's Multiversion concurrency control mechanism, where each modification to the data is saved as a new 'version' of the document. It does however require that the database is compacted which removes unused sections created during updates, it also removes old versions of documents (it does save some metadata for conflict resolution). The amount of history removed is tuneable, the default is to keep 1000 revisions.

5 Discussion

In the previous sections we have discussed and compared the different MapReduce implementations Hadoop, CouchDB and MongoDB according to their performance, applicability, scalability and reliability. In this section we will use these comparisons to conclude what is the best implementation for MapReduce processing.

Hadoop's strong points are that it is currently used in large cluster (over 1.000 nodes) by Yahoo!, proving that it can be used successfully in an enterprise setting. Also the location awareness feature of HDFS allows Hadoop to replicate data to achieve good reliability. Even an outage of an entire rack is possible, without compromising the availability of the data. The downside to using HDFS is the vulnerability of the filesystem by using a single NameNode, as described this is a very critical component for the cluster to function properly.

CouchDB has a strong concurrency model but, as it does not provide a way to scale horizontally by default, relies on third party solutions to provide sufficient processing power. Therefore it can be concluded that the default CouchDB implementation alone is not enough, researching CouchDB Lounge or BigCouch was out of scope. So more research in this area is needed.

MongoDB provides a strong and reliable scaling mechanism through sharding, and provides good data protection by replication. Also the vulnerability of a single NameNode is not present, since the configuration servers provide their own replication scheme.

Based on this we conclude that MongoDB is in general the best MapReduce solution at time of writing. Hadoop nor CouchDB provide the same feature set that is available when using MongoDB.

References

[1] `http://www.mongodb.org/display/DOCS/Sharding+Introduction#ShardingIntrodu%ction-ConfigServers`, Apr 13, 2011.

[2] The Apache Software Foundation. *Cluster Setup - Installation*, 08/17/2010 07:03:47. `http://hadoop.apache.org/common/docs/current/cluster_setup.html`.

[3] The Apache Software Foundation. *File Permissions and Security*, 08/17/2010 07:05:39. `http://hadoop.apache.org/hdfs/docs/current/hdfs_user_guide.html#File+Permissions+and+Security`.

[4] The Apache Software Foundation. *What is Hadoop?*, 10/14/2010 23:35:01. `http://hadoop.apache.org/#What+Is+Hadoop`.

[5] The Apache Software Foundation. *NameNode - Hadoop Documentation*, 2009-10-07 10:08:19. http://wiki.apache.org/hadoop/NameNode.
[6] http://wiki.apache.org/hadoop/HDFS-APIs, 2011.
[7] http://hadoop.apache.org/common/docs/r0.14.4/, 01/23/2008 22:14:39.
[8] http://hadoop.apache.org/hdfs/docs/current/hdfs_user_guide.html, 2011.
[9] T. A. S. Foundation. http://hadoop.apache.org/mapreduce/, 03/20/2011 21:48:10.
[10] R. Ho. http://horicky.blogspot.com/2008/10/couchdb-implementation.html, 2011.
[11] http://couchdb.apache.org/, 2011.
[12] http://guide.couchdb.org/draft/clustering.html, 2011.
[13] http://wiki.apache.org/couchdb/Introduction_to_CouchDB_views, 2011.
[14] http://labs.google.com/papers/mapreduce-osdi04.pdf, 2011.
[15] http://www.mongodb.org/display/DOCS/MapReduce, 2011.
[16] http://www.mongodb.org/display/DOCS/Sharding+Introduction, 2011.
[17] T. White. Hdfs reliability. *http://www.cloudera.com/wp-content/uploads/2010/03/HDFS_Reliability.pdf%*, 2008.
[18] http://en.wikipedia.org/wiki/Hadoop, 2011.
[19] J. Xie, S. Yin, et al. Improving mapreduce performance through data placement in heterogeneous hadoop clusters. *http://www.eng.auburn.edu/~xqin/pubs/hcw10.pdf*, 2010.

High-performance log data analysis using MapReduce

Fernand Geertsema and Erwin Vast

Abstract— The performance of large server clusters can be monitored to optimize the efficiency and response to failures. This can be done by analyzing the log data of all the servers. These log data contain for example the processor usage and memory usage. Existing solutions which process log data in batches are not suitable for server clusters which generate a continuous stream of log data. To analyze this large stream of data a distributed computer setup is needed. One promising solution to analyze large data sets in a distributed setup is the MapReduce framework. In this paper we explain how the MapReduce framework can be used for log analysis to monitor large computer clusters. To use the MapReduce framework in this problem domain of log analysis the following questions arise. Which MapReduce framework should be used for analyzing log data? Which steps need to be included in an algorithm for analyzing log data in a distributed setup? Based on the performance advantages, Apache Hadoop is the best framework for log analysis. In this paper we describe the steps needed for analyzing log data and show the feasibility by applying these steps in an application in a distributed setup. The results show that Hadoop is most useful for large data sets. Dividing the tasks over the different systems takes a minimum amount of time regardless of the size of the data set. Therefore the MapReduce framework in a distributed setup is ideal for pseudo real-time monitoring with a short delay (start-up time plus analysis time) in a large organization with more than 50.000 servers.

Index Terms—MapReduce, Hadoop, high-performance, distributed computing, log data analysis.

1 Introduction

A computer cluster contains a lot of individual servers, each with its own logging facilities. This makes it difficult for server administrators to analyze the overall performance of an entire cluster. Combining log files of the individual servers to one log file can be very useful to get information about the performance of the cluster. The combined log files make it possible to visualize the performance of the cluster and detect problems in the cluster in a short amount of time. However, storing server logs of a cluster for a few days results in data sets of several gigabytes. Analyzing such a huge quantity of data requires a lot of processing power and memory. A distributed system could be the answer to this need for processing power and memory.

One promising solution to analyze large data sets is the MapReduce framework. This is a framework for developing applications which can analyze large data sets on a distributed system. There are already applications available for analyzing large data sets that use MapReduce, like Aster Data [1] and IBM Netezza [2]. These solutions can be used for analyzing web traffic [3] or matching names with identities using fuzzy logic [4]. Another application available on the market for analyzing log data is Splunk [5]. This solution is specifically built for monitoring and analyzing an IT architecture based on various kinds of logs, like application logs and performance logs.

However, this and other solutions process the data in batches, instead of continuous data streams. Batch solutions run for example once a day and are not very well suited for pseudo real-time monitoring. The problem is therefore as follows: *how can the MapReduce framework be used for log analysis to monitor large computer clusters?* Pseudo real-time monitoring requires a MapReduce application that continuously aggregates all log files, which are the base for an average overall performance report. To develop such an application, the following two questions arise.

The first question is: Which MapReduce framework should be used for analyzing log data? There are different implementations available for the MapReduce framework, for example CouchDB, MongoDB, Hadoop, Cassandra and HyperTable. The selection of the most suitable framework for analyzing large data sets will be based on the database type used and the performance of the framework. The second question is: Which steps need to be included in an algorithm for analyzing log data in a distributed setup? In this paper we describe the steps needed for analyzing log data and show the feasibility by applying these steps in an application.

The goal of this paper is to describe how a MapReduce application for log data analysis can be implemented and to show that the MapReduce framework is the right solution for monitoring a large computer cluster.

In the next section, we compare the MapReduce implementations and explain which implementation is the best for log file analysis. Section 3 explains the implementation details and the steps of the MapReduce algorithm. In section 4 we describe how the application is tested, what the results are and a discussion of the research results. Section 5 contains a summary and relates the results to the research question and the general research area.

- *Fernand Geertsema is a MSc. Computing Science student at the University of Groningen, E-mail: f.s.geertsema@student.rug.nl.*
- *Erwin Vast is a MSc. Computing Science student at the University of Groningen, E-mail: e.vast@student.rug.nl.*

2 MapReduce

This section introduces the MapReduce framework which is used for analyzing large amounts of log data. The first subsection gives some general information about this framework. There are different implementations available for MapReduce, which are further discussed in subsection 2. Subsection 3 discusses which of the implementations can be used best for analyzing log data.

2.1 General

MapReduce is a programming model for problems with large data sets that cannot be solved by a single machine. Such problems often are accompanied by data sets of several petabytes. Programs using the MapReduce framework are therefore parallelized and executed on a large cluster of machines, also called nodes [6]. Problems that could be solved by MapReduce are inverse indexing, count word occurrences and distributed sorting [6]. The framework is inspired by the map and reduce function in functional programming languages, however, the functions are not used in the same way [7].

The MapReduce model can be split in two essential functions: the Map function and the Reduce function.

Map: The map function takes an input key/value pair and partitions it into smaller sub-problems and distributes this to worker nodes. Each worker node groups together all intermediate values for the same intermediate key and passes them to the Reduce function.
Reduce: The reduce function accepts an intermediate key and a set of values for that key. It merges these values to create a smaller set of values.

The map and reduce functions are visualized in figure 1. First the input data is sent to the map functions, which maps every key/value pair to a reducer function. The reducer function sorts, groups and aggregates all the values for a certain key. The output of each reducer are the merged values of a key.

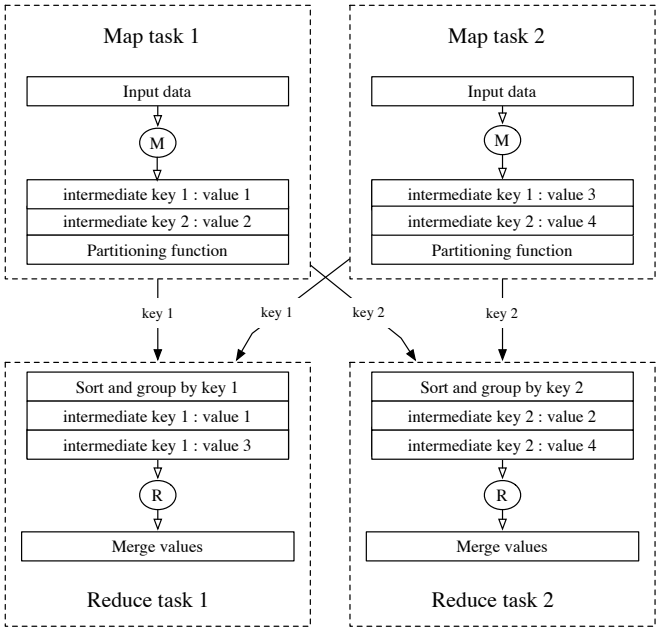


Fig. 1. *MapReduce functions*

If the map functions can be run independent from each other, then they can run in parallel on different computers. The same is valid for the reduce function. An advantage of running the functions on multiple computers is a higher performance of the overall process.

An example problem where the map and reduce functions can be used, is to count the number of occurrences of a word in a large collection of documents [8]. The pseudo-code used for the map and reduce functions in this example is shown in listing 1.

```
1 map(String key, String value):
2     // key: document name
3     // value: document contents
4     for each word w in value:
5         EmitIntermediate(w, 1);
6
7 reduce(String key, Iterator values):
8     // key: a word
9     // values: a list of counts
10    int result = 0;
11    for each v in values:
12        result += ParseInt(v);
13    Emit(AsString(result));
```

Listing 1. Counting words with MapReduce

The map function takes the document name and the document contents and emits for every word the intermediate number 1. The reduce function takes the word as key and a list of occurrences of that word for every document. The function sums the number of occurrences for that word and emits the total amount of occurrences of a word for the complete document collection.

2.2 Implementations

A common setup that can be used with the MapReduce framework is shown in figure 2. This involves a database which supplies the input data and a MapReduce framework which writes the results to the database. Certain types of databases support MapReduce operations,

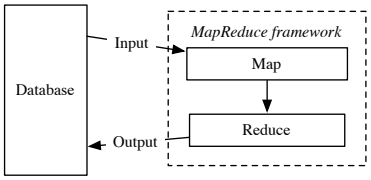


Fig. 2. *MapReduce communication with database*

for example CouchDB and Cassandra [9]. Although the MapReduce framework does not require a database, it is common to use one. The MapReduce implementations that do not use a database require only a file system for storing data. For example Hadoop uses Hadoop Distributed File System (HDFS) for data storage and Google's MapReduce implementation uses the Google File System. However, Hadoop can use the HBase database which runs on top of HDFS. These and other MapReduce implementations can be compared by looking at their functionality, performance, documentation, database type or using the ACID (atomicity, consistency, isolation, durability) properties [10]. The choice for one of these criteria is depending on the structure of the log data.

All machines of a computer cluster have similar types of information to log, for example processor usage, hard disk usage and number of running processes. For that reason, the log data of the machines are structured in the same manner. This structure makes it easier to filter data by information type or selecting the data within a time window. However, these operations require a database, because a database can store the data in an efficient manner. Storing log data in a database prevents that all data from log files have to be read when only a small amount of information is requested. Therefore we compare the MapReduce implementations by database type.

Most of the existing databases that are used are relational databases. In relational databases the data are stored in tables, consisting of rows and columns. The data are managed by the Structured Query Language (SQL). Database types that intentionally do not use SQL, are NoSQL databases. Some advantages [11] of NoSQL databases over relational databases are: better distributable across multiple servers, support for unstructured data and better performance. Log analysis of a cluster is accompanied with very large data sets, that practically cannot be stored on a single filesystem. Therefore the choice is made to use a NoSQL database, that can distribute the data over multiple servers. There are three kinds of NoSQL databases available [11].

The first kind is a key-value database. As the name implies, this database type stores the values which are accessible by a certain key. A key-value database is very useful when the database scheme has to be flexible. Key-value stores also often use cache mechanisms to retrieve data which need frequent access. An example of a key-value store is Amazon's SimpleDB.

Column-oriented databases, the second kind, stores closely related data in a single extendable column. Examples are HBase and Cassandra. An advantage of this type of database is that it is very efficient in doing calculations for a single column, for example averaging all the values of the column memory usage. Column oriented databases are also very useful for distributed storage. A disadvantage of this database type is that it is not flexible. The database structure has to be determined before storing the data and cannot be changed afterwards.

The third kind, a document-based database, stores the data in documents where each document can contain a different number of fields. CouchDB and MongoDB are examples of document-based databases. Document-based databases are useful when the database scheme has to be flexible.

2.3 Which implementation to use

Key-value stores and document-based databases are primarily developed for applications that need to be flexible. However, flexibility is not the primary concern. The log attributes, for example processor and memory usage, are not going to change frequently.

Therefore, the structure of the log data is stored in a predefined matter in the database which is not going to change frequently.

However, performance and distribution are important requirements for log analysis. Column-oriented databases provide these attributes. The flexibility of key-value and document-based databases is less important and therefore the column-oriented database type is the best type for log analysis. The column-oriented database type consists of the following implementations which support MapReduce: Hadoop, Cassandra and Hypertable. Two criteria are important when choosing one of these implementations.

When analyzing logs with MapReduce, read performance is an important criterium. Gigabytes of data have to be read to aggregate all the logs of a computer cluster. Read speed is therefore an important criterium for choosing one of the three implementations. Another criterium is the available documentation of the implementations. To test the algorithm, the MapReduce implementation has to be installed on a distributed system. The success of this task depends on the available documentation created by the developers and other MapReduce users. Based on these two criteria an implementation is chosen.

Hypertable is an open-source implementation based on Google's Bigtable. According to the developers, Hypertable performs better for larger data sets than Hadoop [12]. However, Hypertable's documentation and information about other benchmarks is very limited. Hadoop is more popular, has a larger user community and supports the HBase database. There are many installation guides and answers to common problems, which is very helpful for developing MapReduce applications. Cassandra is slightly faster in writing data than HBase, but HBase is much faster in reading data [13].

The higher read speed is an important advantage of HBase. The popularity and large amount of documentation is an advantage of Hadoop. Therefore, we use the Hadoop MapReduce framework, in combination with the HBase database.

3 Solution

This section describes the implementation to analyze the log data. In the first subsection, an explanation of the Hadoop framework is given. The way the database is structured is explained in subsection two. The last subsection describes the algorithm for the MapReduce framework to analyze log data.

3.1 Hadoop

By default, Hadoop works with HDFS, the Hadoop Distributed File System. This file system is created by the Hadoop user community to save files in a distributed environment, where the data are stored on multiple servers. Storing data on multiple servers has as advantage that the analysis of the logs can be spread over the servers.

The distributed Hadoop system can be split in two parts [14]: the processing part and the storage part. The processing part focuses on the MapReduce tasks and distributing the tasks among the several nodes that are available. The storage part is for retrieving the data which are needed for processing and writing the results. The processing and storage services are distributed over master nodes and slave nodes. The master nodes are responsible for distributing the tasks to the slaves. The slave nodes do the actual processing.

The storage part of the Hadoop system has the following services:
NameNode: this service manages the namespace, file system and access control. There is one primary and one secondary NameNode in each cluster. The secondary NameNode is for fault-tolerance.
Datanode: this service holds the file system data, and can be replicated over multiple DataNodes. If the Hadoop cluster contains only one DataNode, the data cannot be replicated.
The processing part of the Hadoop system has the following services:
JobTracker: this service assigns the different tasks to the slave nodes. Each cluster has one JobTracker.
TaskTracker: this service carries out the map and reduce tasks. It is recommended to have multiple TaskTrackers to get benefits from the distributed setup.

The structure of the Hadoop system is illustrated in Figure 3. It shows three blocks which represent the servers in the system. These servers run multiple services. For example the master server is a Jobtracker and a NameNode. This server manages the different MapReduce tasks and is also in control of the filesystem of the cluster. The other servers work as slaves for the master. Each slave is both a TaskTracker and a DataNode, because the TaskTrackers need the data from the DataNode when running MapReduce [15]. Each slave therefore processes the MapReduce tasks and stores data. The master server can split his tasks and run the JobTracker and the NameNode on two separate servers. This will improve the performance of the analysis while executing large MapReduce tasks. Most Hadoop applications run on a distributed system, but it is also possible to run all processes on a single machine, also called a single computer setup. This is mostly used for development and testing purposes which involve smaller MapReduce tasks.

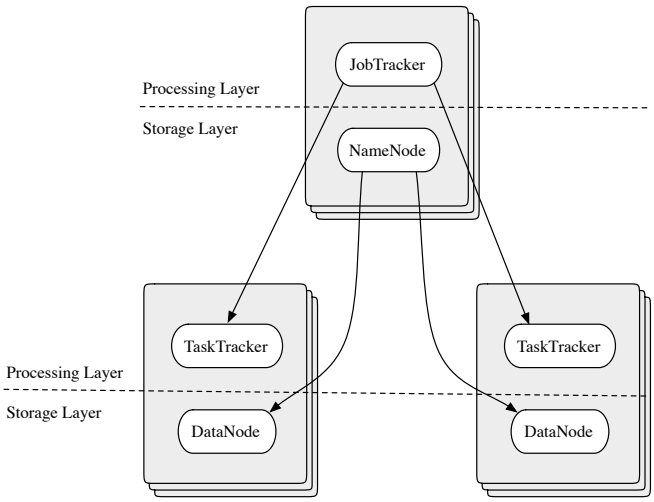


Fig. 3. *Hadoop process structure*

3.2 Log data in HBase

To get knowledge about the performance and statistics of a cluster, there must be some logging done by the individual servers. These logs can be stored to log-files which have to be retrieved by a logging server or will be sent directly to the logging server. The Simple Network Management Protocol (SNMP) can be used for sending logs [16] to another server. This protocol is developed specifically to retrieve performance of servers from a central point. Normally a SNMP-server is a good way to collect log data from a cluster. However, there are no log-files from a large cluster publicly available for analysis.

Therefore, for the test in section four the log-files are created by a Java application, which generates pseudo random statistics, further called attributes, and writes these attributes to a database. These attributes describe the current state of a server and include the following twelve values of a server: "cpu usage", "kernel memory commit", "physical memory commit", "physical memory swap", "processes", "threads", "handles", "hard disk read", "hard disk write", "network usage", "uptime" and "round trip time". Every minute that a server runs, simulated by the Java application, it creates a log file. Our simulation creates up to 1.440 log-files per server for each day. As will be described in chapter four, large companies can have more than 50.000 servers. This can lead to 72.000.000 log-files per day. This number of log-files results in 86 GB of data per days.

The log data created by each server are sent to the HBase database. This database system is column-oriented, therefore all data are stored in columns and not in the traditional row system. This gives performance gains when there are many rows (values) needed and only a few columns are affected. HBase has its database built in the same way as

Google's BigTable [17]. The database structure used to store log data is shown in figure 4.

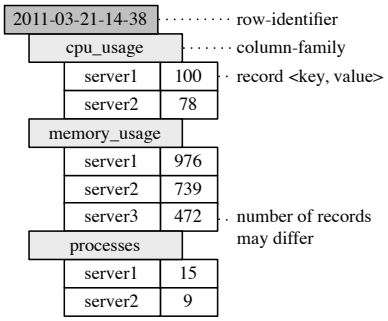


Fig. 4. *Database structure*

The row-identifier contains the date, because this makes it easier to retrieve the data from a specific time span. If we would, for example, use the server name as the row-identifier, we would have to retrieve all the data for a single server. This is not very efficient, because we only want data from all servers from a specific time span. For monitoring applications, such time windows would be several minutes wide.

The column-family contains the attribute (e.g. cpu usage) and not the name of the server. Using the name of the server prevents that machines can be added to or removed from the computer cluster. In section 2 it was described that column-oriented databases are not flexible. Therefore, the column-families cannot be changed after initializing the database. As a result, the column-families are not constructed with the names of the server. This makes it possible to change the amount of servers after initializing the database.

The record contains the statistical data. The key of the record is the name of the server (e.g. server1). The value is the data of the attribute (e.g. 100). Each column-family contains the attributes for that family (e.g. cpu usage) for all machines of the cluster.

3.3 MapReduce tasks

As explained in section 2.1, the MapReduce framework depends on two fundamental functions: the map and reduce functions. For the log analyzer these functions are as follows.

Map function: The input of the map function are multiple rows from the database. Each row stores the log of one timestamp, for example, one minute. In this case a time window of five minutes is used. Because these five minutes include the attributes of all the servers that are monitored, this leads to many gigabytes of data. Therefore the data are split at each attribute, like "cpu usage" or "network throughput". An intermediate result is created. This intermediate result has a key (e.g. cpu) and a value (e.g. 100). The key of this intermediate result is the attribute. The value of the intermediate result is the value of the attribute. This intermediate result is sent to the reduce function.
Reduce function: All the same keys (e.g. cpu, memory) of the map function are mapped to a reducer function. This leads to one reducer per attribute. This reducer collects all the values of that attribute for the last five minutes. The reduce function sorts the incoming intermediate result by the keys. The values of the attributes are extracted and averaged. This computed average is then saved to the database. In this way the attributes of the servers for a time window of five minutes are stored.

Both functions are listed in pseudo code in listing 2. To visualize the MapReduce process, the flow of the map and reduce functions is displayed in figure 5. The map function is visualized in Map task 1 and 2. These tasks receive each one minute of log data. In this example the attributes cpu and memory of two servers are processed. Map task 1 and 2 decode each a minute of log data and extract the attributes. These attributes are then put into an intermediate result and sent to the reducer. These messages have as keys "cpu" and "mem". For example the "cpu" attribute of server 1 is shown in the block "(s1) cpu : v1". The cpu attribute for the second minute is shown in block "(s1) cpu : v5". The intermediate results with the key "cpu" are received by Reduce task 1. This reduce function collects all the cpu attributes. The memory attribute goes to Reduce task 2. The reducers calculate the sum of all the values received. This sum is then divided by the number of intermediate results received, which gives an average value. For the cpu attribute the values v1, v2, v5 and v6 are added up and divided by four. This gives the average cpu usage of two minutes. The average cpu usage is stored in the database with the timestamp and the type of attribute (i.e. cpu).

```
1 map(RowData row):
2   // row: contains a minute of log data from the
        database
3   String familyName;
4   // familyName: is the attribute (e.g. cpu)
5
6   for each column c in row:
7     familyName = new String(c.getFamilyname());
8     int value = ParseInt(c.getValue());
9
10    // Send attributes to the reducers, with key=
          familyName
11    EmitIntermediate(familyName, value);
12
13 reduce(String key, Iterator values):
14   // key: a familyName
15   // values: a list of values of the familyName
16   String timestamp = job.getName();
17   long sum = 0;
18   int numberOfValues = 0;
19
20   for each v in values:
21       sum += ParseInt(v);
22       numberOfValues++;
23
24   int average = (sum / number_of_values);
25   saveAverageInDatabase(timestamp, key, average);
```

Listing 2. Counting words with MapReduce

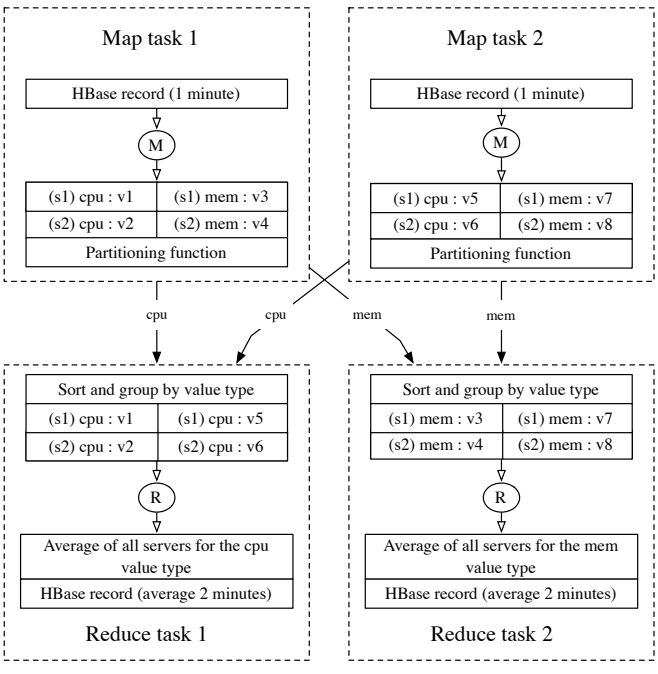


Fig. 5. *MapReduce process structure for log analysis*

4 TESTS

In this section the implementation is tested. The first subsection describes the test setup that is used for testing the application. The results of the tests are described in subsection two. The last subsection contains a discussion over the test results.

4.1 Setup

An important aspect of a potential solution is to test it. The main quality attribute for the implementation is performance. To test the performance, the application is tested on a distributed system and a single computer setup. For running a distributed system, we use the Amazon Elastic Compute Cloud, also called Amazon EC2. This web service of Amazon makes it possible to create virtual machines remotely [18]. The number of virtual machines in use depends on the load at a certain moment in time. Our distributed system consists of one master node and four slave nodes, therefore five virtual machines are needed. The single computer setup consists of one node which runs all the required services that were described in section 3.1.

Amazon offers several types of machines, each with its own hardware specifications. For this test, we used the Large Instance type. This type [19] consists of 7.5 GB memory, 4 EC2 Compute Units, 850 GB allocated storage and high I/O performance. One EC2 Compute Unit is comparable to a 1.0-1.2 GHz 2007 Opteron processor. All EC2 instances are located in the same region to increase network speed. The single computer setup runs on a machine with 4.0 GB memory, 500 GB allocated storage and a 2.4 GHz dual core processor.

For large companies with many servers, the performance of the cluster is essential because it affects many customers. Statistics [20] show that large companies like Google, Microsoft and Amazon have more than 50.000 servers. In our test case we therefore analyze the logs of 50.000 servers to reflect these large computer cluster sizes. To simulate this scenario, the following two applications are used for the test.

The simulator simulates a cluster and stores the data in HBase. It stores the log data of 50.000 servers each minute. This is a constant stream of log data.

The log analyzer applies the MapReduce functions that were described in section 3. The log analyzer retrieves the log files from the database, combines these, and stores the combined logs in a separate table in HBase. To create pseudo real-time monitoring, the application retrieves the data for a given time window. The time window is five minutes for this test. As a result each MapReduce function analyzes a collection of 250.000 log files each with twelve attributes (e.g. cpu and memory usage). The log analyzer retrieves and analyses every five minutes the new logs the simulator stored in the database during those five minutes.

4.2 Results

For this test five different sizes of log data sets have been analyzed using the MapReduce framework in a distributed computer setup and in a single computer setup. Table 1 shows the results of this test. Five minutes of log data from 50.000 servers leads to 250.000 log files each with twelve attributes. The log data sets for the test cases have the following sizes: 250.000, 500.000, 1 million, 2 million and 4 million server logs. The MapReduce framework is used in this test for the log analysis of each log data set. The test is performed three times per log data set to get an average analysis time.

Figure 6 shows the performance results as a graph. The square points represent the single setup and the triangle points show the results of the distributed setup. The single setup has only three points, because the tests of the single setup could not finish for 2 million and 4 million server logs. The last two test cases are too large to process for the single setup. When analyzing more than 1 million logs, the hard disk is not fast enough to read the data for analysis and write the log data from the simulator. Both processes get increasingly more behind. Therefore the last two test cases are never completed for the single setup. The results show that in each test case the distributed setup can process the log data.

Table 1. Performance results

Number of server logs	Size data set (megabytes)	Average analysis time (seconds) (distributed setup)	Average analysis time (seconds) (single setup)
250.000	218	52	131
500.000	437	67	259
1.000.000	1101	99	625
2.000.000	2862	146	*
4.000.000	4770	247	*

* The test could not finish

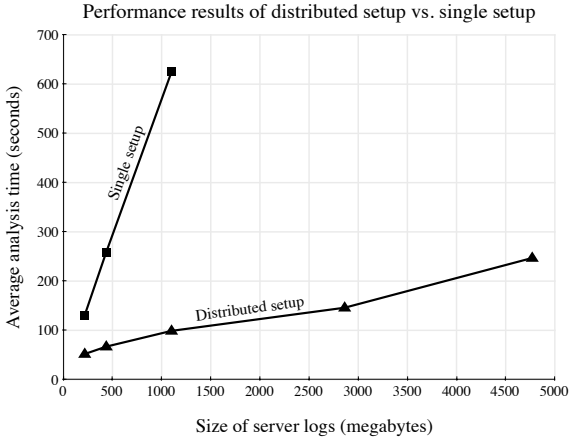


Fig. 6. *Performance results*

The results show also that the distributed setup can process the largest test case under five minutes. This proves that the distributed setup can be used for pseudo real-time analysis of large log data sets with only a short delay. The single setup is only able to run the first two test cases in less than five minutes. The third test case finished in more than 10 minutes. This shows that a single setup is not suitable for pseudo real-time monitoring of large log data sets.

The above results prove that the MapReduce framework can be used for analyzing log data to monitor large computer clusters. For each log analysis the MapReduce framework has some start-up time, therefore a time window of the log analysis has to be larger than the minimum time needed to analyze the log data. Therefore the MapReduce framework in a distributed setup is ideal for pseudo real-time monitoring with a short delay (start-up time plus analysis time) in a large organization with more than 50.000 servers.

4.3 Discussion

The tests were performed on a log data set with twelve attributes (e.g. cpu and memory usage). These attributes represent normal performance data, which are common in log files. The number of monitored attributes can differ per analyzed computer cluster. Analyzing more attributes will lead to an increase in the size of the log data and a higher average analysis time. To manage this increase in analysis time, the distributed setup can be expanded by using more slave nodes to process the log data.

The distributed setup used for the test was created by using the Amazon EC2 service. With this service the user has to pay per hour for each virtual machine. The continuous analysis of log data requires that the distributed setup is always online. The Amazon EC2 service is provided to give users a flexible way to cope with changing needs for processing power. Because the log analysis is a continuous process, costs could be reduced by creating a distributed setup locally instead of using Amazon EC2. Another advantage of running a distributed setup locally is that the log data remain on the servers of the company.

5 CONCLUSION

The question that was asked in the beginning of this paper was "How can the MapReduce framework be used for log analysis to monitor large computer clusters?" To answer this question the subquestions "Which MapReduce framework should be used for analyzing log data?" and "Which steps need to be included in an algorithm for analyzing log data in a distributed setup?" have to be answered first.

For analyzing log data with the MapReduce framework, the Apache Hadoop framework is a favourable choice based on the support for the column-oriented database HBase. By using this database in combination with the Hadoop framework a powerful toolset is created that has a high performance. The Hadoop framework is better documented than other MapReduce frameworks.

The steps needed for analyzing log data include distributing these log data over multiple systems and sorting these by attribute (e.g. cpu usage). After sorting these log data, the values of these attributes are averaged. This creates an average value per attribute for the whole cluster.

By answering these subquestions an implementation with the use of the Hadoop framework can be built. The built implementation is tested on a distributed computer setup of five machines and a single computer setup. The results show that for large sizes of log data the single computer setup could not analyze them. The distributed setup analyzed all the different test cases of which the largest log data set consists of 4 million server logs and finished each test case in less than five minutes. This makes the MapReduce framework in a distributed setup ideal for pseudo real-time monitoring with a short delay (start-up time plus analysis time) in a large organization with more than 50.000 servers.

The log data that was used to test the implementation originated from a simulator. For future research, real log data of an existing computer cluster can be used. Another improvement is to test with larger log files to further analyze the scalability of the implementation.

REFERENCES

[1] Aster Data. *Big Data Analytics, MapReduce for High Performance In-Database Analytics, Deep Data Mining*, March 2011. http://www.asterdata.com/.

[2] IBM Netezza. *Data Warehouse Appliance, Data Warehouse Appliances, and Data Warehousing from Netezza*, March 2011. http://www.netezza.com/.

[3] Aster Data. *MySpace.com Scales Analytics for All of Its Friends*, 2009. http://www.asterdata.com/resources/assets/cs_Aster_Data_4.0_MySpace.pdf.

[4] IBM Netezza. *M3Name Searching with Fuzzy Logic Using Netezza OnStreamTM Analytics*, 2009. http://www.netezza.com/documents/NET7032_MTI_DS_6.pdf.

[5] Splunk. *Operational Intelligence, Log Management, Application Management, Security and Compliance*, 2010. http://www.splunk.com/.

[6] Google, Inc. *MapReduce: Simplified Data Processing on Large Clusters*, 2004. http://labs.google.com/papers/mapreduce.html.

[7] Wikipedia. *MapReduce*, March 2011. http://en.wikipedia.org/wiki/MapReduce.

[8] Communications of the ACM, vol. 53, no. 1. *MapReduce: A Flexible Data Processing Tool*, January 2010.

[9] *NOSQL Databases*, March 2010. http://nosql-database.org/.

[10] Wikipedia. *ACID*, 2010. http://en.wikipedia.org/wiki/ACID.

[11] Computer, vol. 43, no. 2, IEEE. *Will NoSQL Databases Live Up to Their Promise?*, 2010. http://www.leavcom.com/pdf/NoSQL.pdf.

[12] Hypertable. *Hypertable vs. HBase Performance Evaluation Test*, 2010. http://www.hypertable.com/pub/perfeval/test1.

[13] *Benchmarking Cloud Serving Systems with YCSB*, June 2010. http://research.yahoo.com/node/3202.

[14] Brandeis University. *Hadoop Cluster Setup*, 2008. http://pages.cs.brandeis.edu/~cs147a/lab/hadoop-cluster/.

[15] Hadoop. *Cluster setup*, 2010. http://hadoop.apache.org/common/docs/current/cluster_setup.html#Slaves.

[16] Wikipedia. *Simple Network Management Protocol*, 2011. http://nl.wikipedia.org/wiki/Simple_Network_Management_Protocol.

[17] Google. *Bigtable: A Distributed Storage System for Structured Data*, November 2006. http://labs.google.com/papers/bigtable.html.

[18] Amazon. *Amazon Elastic Compute Cloud (Amazon EC2)*, 2011. http://aws.amazon.com/ec2/.

[19] Amazon. *Amazon EC2 Instance Types*, 2011. http://aws.amazon.com/ec2/instance-types/.

[20] Datacenter Knowledge. *Who Has the Most Web Servers?*, February 2011. http://www.datacenterknowledge.com/archives/2009/05/14/whos-got-the-most-web-servers/.

Context Inconsistency Management in Pervasive Systems

Samuel Esposito Alexander Jurjens

Abstract—Thanks to the pervasive computing paradigm more and more computer systems in utility buildings and industry are context-aware. They use a representation of the world they operate in to reduce the human-computer interaction necessary for their operation. Unfortunately reasoning based on contexts is not without flaws and context inconsistencies are the main reason for context-aware applications' incongruous behavior. Context consistency management is not adequately studied in existing literature and approaches for detecting and resolving context conflicts are not suited for pervasive computing [3].
In this paper we present two complementary approaches for improving the mitigation of context inconsistencies. First we present partial constraint checking for timely identifying context inconsistencies at runtime. An extra constraint layer is added to the traditional ontology based context model and conflicts can be detected by locally checking partial constraints in the ontology. This dramatically improves performance compared to iterative evaluation of an entire ontology [3]. Secondly we discuss the extension of the traditional ontology model with context life cycles to more accurately represent the environment of context-aware applications. This information can then be used to estimate the relative reliability of contexts in a conflict set and discard the contexts with lowest reliability [2]. Apart from resolving context conflicts it is also possible to represent inconsistencies into the context model itself [5]. In this paper we present a case study in which we explore the possibilities of incorporating inconsistencies into context models using fuzzy ontologies.

Index Terms—Pervasive Computing, Context Ontology Model, Partial Constraint Checking, Context Lifecycle, Context Inconsistency Resolution, Fuzzy OWL.

1 Introduction

In pervasive computing applications use contexts to represent their changing environment and adapt their behavior to it. Due to environmental noises contexts can be imprecise or incorrect, resulting in context inconsistencies. These inconsistencies may set a pervasive application in a wrong state or misadjust its behavior. Therefore it is important to timely detect and resolve context inconsistencies.

In the next chapter we start with an overview of the existing solutions for handling context inconsistencies and argue why these solutions do not meet the requirements of pervasive applications. Next we present Partial Constraint Checking: an algorithm for constraint checking on huge sets of contexts that is far more efficient and effective than the existing approaches. In addition we discuss a Context Inconsistency Resolution algorithm that allows for resolving conflicting context sets without human intervention. Finally we propose a whole new approach to context inconsistencies: incorporating inconsistencies in to the context model by using fuzzy ontologies. We show that this approach makes conflict detection and resolution superfluous and has a lot of potential for an efficient implementation using Partial Probability Calculation.

2 Related Work

Pervasive or ubiquitous computing is a fast-developing discipline that has been receiving increasing attention from both researchers and software developers [3]. In the past decade, many context-aware systems have been developed, ranging from smart room environments to warehouse and supply chain management systems. A lot of effort has been put into building middleware infrastructures that handle vast amounts of sensory data and extract the context information relevant for pervasive applications. Examples of such systems are CoBrA [4] and CORTEX [1]. Various modeling approaches have been proposed for capturing context information, of which the ontology based context model appears to be most promising for most pervasive applications [2].

Context management for consistency however has not been adequately studied in the existing literature. None of the studies on context-awareness discusses a way for detecting context inconsistencies for reliable pervasive computing [3, 2]. Even though other disciplines as artificial intelligence and software engineering conducted related research, it does not provide adequate support for context inconsistency detection in ubiquitous computing. In addition the strategies proposed in literature for resolving context conflicts are not suited for pervasive computing. Some are based on assumptions that may not apply to general pervasive environments. Other require human participation for conflict resolution, which is usually expensive and slow for pervasive computing [3]. Finally no research has been done on the potential of fuzzy ontologies to represent inconsistencies in the environment or the perception of this environment instead of trying to resolve them [5].

In this article we aim at putting a milestone for context management by presenting an efficient inconsistency detection algorithm based on a constraint language extending the traditional ontology based context model [3]. In addition we put forward a conflict resolution algorithm which is based on a context reliability heuristic [2]. Finally the use of fuzzy ontologies representing context inconsistencies as a promising alternative to conflict resolution is explored.

3 Partial Constraint Checking

Constraint checking techniques have been extensively studied in software engineering. Existing constraint checking algorithms focus on checking software artifacts that do not change rapidly or frequently [3]. Context-aware applications require more efficient algorithms because they use a huge set of contexts, which can change very rapidly and frequently (in the range of milliseconds). An inefficient software solution for this does not only require massive computing power, but interestingly also induces a higher inconsistency detection miss rate. Because of the computing delay conflicting contexts slip through the context buffer before they are detected by the software [3]. One example of an inefficient approach is non-incremental checking: whenever there is a change in the set of software artifacts these artifacts are each checked against the entire set of consistency constraints to find out all detectable inconsistencies. An improvement to this would be incremental checking: only a subset of all constraints are checked, namely those that are affected by the specific change in the artifact set. But Xu *et al.* made the most substantial progress by replacing the traditional entire constraint checking approach by partial constraint checking based on a consistency computation tree [3]. Their idea is that constraints can be represented as trees with nodes for logical operations and leaves for specific properties of contexts or context sets. Whenever a context is added to or deleted from a context set in time, the branch corresponding to this context can respectively be added to or deleted from the tree (see Fig. 1). Because intermediate values are retained in the tree nodes after calculation, they can be reused whenever the tree changes. More specifically, when a branch is

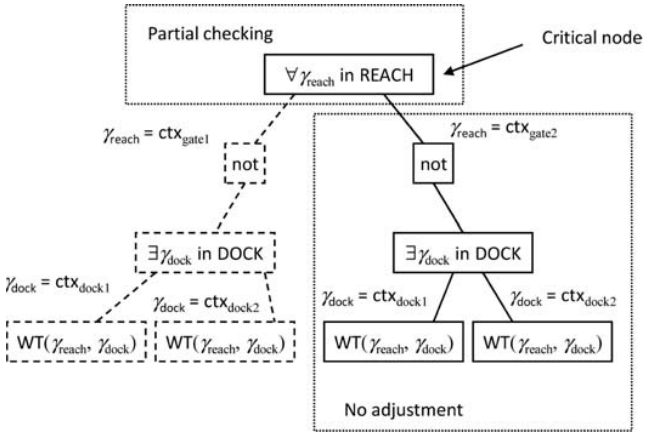


Fig. 1. Consistency Computation Tree: branches can be added or deleted to represent changes in a context set. Source: [3]

added, only the values of the new branch itself and of the nodes from the branch top to the tree root need to be calculated. When a branch is deleted, only the values for the nodes from the branch top to the tree root have to be recalculated.

With their partial constraint checking algorithm Xu *et al.* attained a time complexity between $O(n)$ in the worst case and $O(1)$ in the best case when a context is added to the set and $O(1)$ when a context is removed. Compared to traditional constraint checking with an overall complexity of $O(n)$, this is a dramatical improvement. In their experiments Xu *et al.* showed their performance is 15 times better than the traditional approach and in a case study the inconsistency miss rate dropped from 52.2% in traditional checking to 0.1% with partial checking [3].

4 Context Inconsistency Resolution

Now that we have an efficient method for detecting inconsistencies, the task of resolving the conflicts remains. As discussed above the strategies for conflict resolution in literature are not well suited for pervasive computing because their assumptions do not apply or they require human participation [3]. What is really needed is an algorithm that in the case of a conflict between two or more contexts decides which context has the highest reliability and retains that context. Bu *et al.* show us that this is possible through extending the ontology based context model with additional information about the context's status and temporal properties [2]. More specifically they propose an algorithm that retains for every conflict set the context with the highest *relative frequency*: frequency of context updates relative to update interval and context age (see Fig. 2). This is based on the assumption that

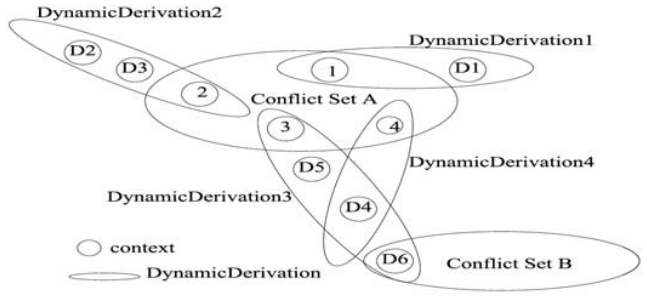


Fig. 2. Context inconsistency resolution in action. Source: [2]

contexts that are most stably perceived by a system are most likely to be correct. This assumption is applicable to most of the environments in which pervasive systems run and is applied in many domains as a domain specific approach [3].

5 Fuzzy Ontologies

In pervasive computing traditionally first order logic is used to model the environment of an application. This results in a so called ontology based context model. Every observation that does not exactly fit in this model results into a conflict. This conflict can lead to misadjusted behavior in applications, unless it is timely detected and resolved [3]. Because it is impossible to perfectly model the unpredictable environment in which context-aware applications run, it would be nice if the context model was more resilient to unexpected observations. This resilience can be obtained by using a fuzzy ontology: a stochastic model representing lower level contexts (for instance sensory data) and their probabilities in the presence of higher level contexts (for instance user activities). The most simple approach to this would be using a hidden Markov Model[1]: a statistical model of the environment with unobserved states (the higher level contexts) in which future states only depend on the present state. The use of such a model is illustrated in the case study below.

5.1 Case Study

Suppose we want to build a context-aware application that has a certain notion of teacher's activities to assist them in the best possible way (for instance automatically opening slides when a lecture starts). In the traditional approach we could build an ontology specifying where lectures take place and that a lecture starts when the teacher appears at the lectern. If now an unpredicted observation is made (sensor cross read or teacher momentarily leaves the room), an inconsistency occurs that has to be resolved for the application to proceed with its normal operation. In the fuzzy ontology approach these unpredictable events can be modeled so that the ontology is more robust and always yields the most apt interpretation of the environment without the need for conflict resolution. A simplified hidden Markov model for this ontology is depicted in Fig. 3. As the numbers in this model indicate, the

Fig. 3. Fuzzy ontology specifying a teacher's activities and the probabilities of different locations in the presence of these higher-level context.

most probable location of a teacher while lecturing is at the lectern (L). In the aisle (A) is less probable and at the office (O) is very unlikely. For the activity of performing other work (denoted as Working in the model), the probabilities are the other way around. This fuzzy ontology model can now be used to calculate the relative probability of each higher-level context (c.q. Lecturing and Working) based on a set of low level contexts observed within a certain time window. A possible set of observations could be $[L_1, L_2, L_3, A_1, A_2]$: the teacher was observed three times at the lectern and twice in the aisle in this time window. The probability of the Lecturing activity can now be calculated by multiplying the probabilities of the observations in the presence of this higher level context[2]: $pLecturing = .80 * .80 * .80 * .15 * .15 * c_1 = .01152 * c_1 = 0.9998$. The probability for Working can be calculated in a similar way: $pWorking = .05 * .05 * .05 * .15 * .15 * c_1 = 0.000002813 * c_1 = 0.0002$. The probabilities indicate that the teacher is still lecturing, even though the teacher being observed in the aisle conflicts with being observed at the lectern. Another example of observations could be $[L_1, A_1, A_2, O_1, O_2]$: the teacher is observed at the lectern once, twice in the aisle and twice at the office. The probabilities for Lecturing and Working are calculated as follows:

[1] http://en.wikipedia.org/wikiHidden_Markov_model

[2] We multiply the results with a constant c to obtain a valid probability value in the end.

$pLecturing = .80 * .15 * .15 * .05 * .05 * c_2 = 0.000045 * c_2 = 0.059$ and $pWorking = .05 * .15 * .15 * .80 * .80 * c_2 = 0.00072 * c_2 = 0.941$. The probabilities now indicate that the teacher is doing work other than lecturing, even though the observation of the teacher at the lectern is definitely conflicting with the observations at the office.

5.2 Results

The case study shows that a fuzzy ontology allows us to always define the most probable higher-level context given the observations, regardless of any inconsistencies in the observed context set and without the need for conflict detection and resolution.

Another observation we can do is that the hidden Markov model for a fuzzy ontology has a tree-like structure. This structure allows for very efficient incremental Partial Probability Calculation (PPC). The PCC principle works as follows: the probability of a high-level context is calculated by multiplying the probabilities of its children contexts. When a new context is added to or removed from the observation set and thus from the tree, only the probabilities of the observation's ancestors in the tree up to the root have to be recalculated. In case of adding a context, the probability of an ancestor in the tree can be recalculated by simply multiplying its previous probability with the probability of the new or updated child. In case of removing a context, the probability is divided by the probability of the new or updated child. Because the recalculation only takes place in part of the tree and allows us to preserve previous calculations for later use, it is partial. And because this recalculation only happens in trees that contain the added or removed context as a leaf, the calculation is incremental. We illustrate the PPC process using an extension of the hidden Markov model from the case study (See Fig. 4). In this figure the

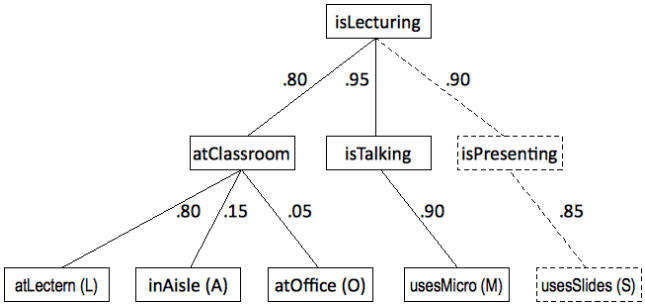


Fig. 4. Fuzzy ontology specifying the Lecturing activity and the probabilities of different contexts in the presence of this higher-level context.

usesSlides context is added to the observation set at some point. This change in the context set requires us to recalculate the probabilities of the higher-level contexts. For this we use PPC: we recalculate only the values of the ancestors if the usesSlides context (c.q. isPresenting and isLecturing). The probability of isPresenting is $.85 * .90$, because the usesSlides context is its only child. The probability of isLecturing has to be recalculated. This can be done by simply multiplying its previous probability with the probability of the new child isPresenting: $isLecturing = 0.0004104 * .90 * .85 = 0.000313956$[3].

The time complexity of the PPC principle is related to the height of a hidden Markov tree, because recalculations only take place on the path from a leaf to the root. Since this height remains constant at runtime, the time complexity of adding or removing a context is $O(1)$.

It is possible to extend the hidden Markov model used in the case study by specifying start and transition probabilities for the higher-level contexts. With this extension the probability of a higher-level context does not solely depend on the contexts in the observation set, but also on the previous most probable higher-level context.

[3] Note that this value has to be multiplied with a constant in order to get the real probability value

6 Discussion

In this article we started with presenting partial constraint checking: an efficient algorithm for timely detecting inconsistencies in huge sets of contexts which change frequently. Partial constraint checking was 15 times more efficient than the traditional greedy approach and reduced the conflict detection miss rate from 52.2% to 0.1%. This algorithm is clearly a dramatic improvement of the traditional approach and vital to any context-aware application that has to handle frequently generated contexts.

After that we shed light on ways to resolve the detected conflicts and discussed a context inconsistency resolution algorithm which estimates the reliability of conflicting contexts and only retains the most reliable context from every conflict set. This algorithm allows for timely conflict resolution at runtime without the need for human participation. The algorithm is a valuable addition to the research on conflict resolution in pervasive computing, but when applying it one has to verify the aptness of the heuristic used for context reliability estimations in the domain at hand. In particular the assumption that contexts with the highest relative frequency are most reliable has to apply in the domain.

Finally the use of fuzzy ontologies for modeling the environment of context-aware applications seems promising. By incorporating possible inconsistencies in the model, the need for context inconsistency management vanishes, vastly reducing the complexity of the used middleware. Fuzzy ontologies always yield the most probable higher-level contexts, regardless of the inconsistencies in the observed context set. In addition the calculation of the probabilities in a fuzzy ontology has great potential for optimization by using incremental Partial Probability Calculation. When using fuzzy ontologies one however has to take into account the required training of the system for obtaining well defined probabilities in the context model.

7 Conclusion

In conclusion we can say that the partial constraint checking algorithm can dramatically improve the performance of context-aware applications where context consistency management is needed. And with this performance improvement we get an impressive reduction of missed context inconsistencies for free. The context inconsistency resolution algorithm is a good solution in most domains for timely resolving context conflicts without human intervention. The fuzzy ontology approach seems a promising alternative to traditional context modeling, but needs extensive further research in order to determine its real value for context-aware applications.

References

[1] G. Biegel and V. Cahill. A framework for developing mobile, context-aware applications. In *Pervasive Computing and Communications, 2004. PerCom 2004. Proceedings of the Second IEEE Annual Conference on*, pages 361 – 365, 2004.

[2] Y. Bu, S. Chen, J. Li, X. Tao, and J. Lu. *Context Consistency Management Using Ontology Based Model*, volume 4254, pages 741–755. Springer Berlin / Heidelberg, 2006.

[3] W. C. Chang Xu, S.C. Cheung and C. Ye. Partial constraint checking for context consistency in pervasive computing. *ACM Transactions on Software Engineering and Methodology*, 19, 2010.

[4] H. Chen, T. Finin, and A. Joshi. Semantic web in the context broker architecture. In *Pervasive Computing and Communications, 2004. PerCom 2004. Proceedings of the Second IEEE Annual Conference on*, pages 277 – 286, 2004.

[5] H. Kong, G. Xue, X. He, and S. Yao. A proposal to handle inconsistent ontology with fuzzy owl. *Computer Science and Information Engineering, World Congress on*, 1:599–603, 2009.

The Role of Standardized Web Services in Electric Utility Control Center Applications Integration

Divya .S. Avalur*

Abstract— The power system operation applications of the Electric Utility Control Center such as *Energy Management Systems (EMS) / Supervisory- Control-And-Data-Acquisition (SCADA)* are installed as stand alone systems.The power system operations applications are highly specialized and complex in nature. Due to this, the integration of power system operation applications with business applications of the Electric utility becomes quite challenging. The applications at business end of the electrical utility control center are well integrated using standardized web services and Service Oriented Architecture(SOA) middleware. The paper details out the standardized web services and its role in the integration of power system operation applications using International Electrotechnical Commission(IEC) standards. In particular, the paper concentrates on IEC 61970 and IEC 61850 standards. It further aims to explore various aspects of standardized web services like implementation and testing using a case study.

Index Terms—Generic Interface Definition(GID), Common Information Model(CIM),IEC 61970, IEC 61850, EMS/SCADA, SOA

1 Introduction

An Energy Management System (EMS) is a controlling and monitoring system of computer-aided tools allowing operators of electric utilities to monitor, control and optimize the performance of the generation and/or the power system. The Supervisory-control-and-data-acquisition(SCADA) system is used to data gathering and monitoring of automated sytems, their conditions and control. These services when implemented in an open-source fashion have certain specific advantages compared with other sources [1]. They enable the reduction of costs in upgrading, application integration, application decoupling and ease of development. The integration of power system operations applications of the Electric Utility Control Center such as Energy Management Systems (EMS)/ Supervisory- Control-And-Data-Aquisition (SCADA) with business applications of the electric utility like finance, customer information systems and asset management is quite a challenging task. The approach used in earlier technologies consisted of stand alone systems with highly specialized applications. They interfaced with the power systems using communication based protocols. The major drawback in earlier technology was its inability to integrate the power system applications with business applications of utility. To eliminate this problem, International Electrotechnical Commission (IEC) has come up with a set of standards that enable web services and Service-Oriented-Architecture (SOA) middleware technology for the integration of systems. Due to these services, the cost of infrastructure, programming effort and complexity has decreased dramatically [2].

The following is the organisation of the paper. In Section 2, we shall discuss about the web services, its fundamental components and demerits of the normal web services. In Section 3, we discuss various aspects of International Electrotechnical Commission (IEC) standards, give a description of the selected IEC standards and later discuss the need for the standardized web services. In Section 4, we discuss various aspects of IEC 61970 standard such as its components and benefits of implementing these components together. In Section 5, we discuss the application of a particular component of IEC 61970 in Energy Management Systems (EMS). In Section 6, we present a case study of application of the standardized web services using open source implementation.

2 Web Services

We make use of the web services in our day to day environment by describing product or service using web pages. Semantic web services are the improvements of the known concept of the web services.

* *Johann Bernoulli Institute of Mathematics and Computing Sciences, University of Groningen, The Netherlands. e-mail: d.avalur@student.rug.nl*

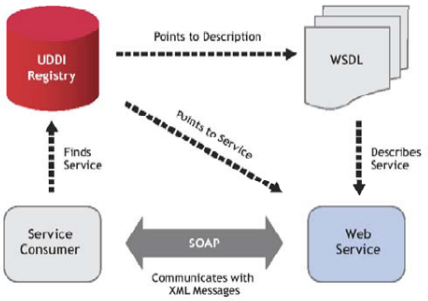


Fig. 1. Webservice Architecture[2]

The term *web services* is used to describe many activities as mentioned above, and hence is highly context-dependent. In this paper, we are concerned about the application integration of the electric control utility center using web services. As far as application integration is concerned, web services are referred to as structured architectures on which various computer applications can be integrated with each other using eXtensible Markup Language (XML) over network architecture like World Wide Web(WWW). The main goals of the web services is to build an integration architecture that not only provides reuse of existing infrastructure but also allows the clients to utilize the services and data supported by the server. The web services also provide framework for integration development using tools like Java, .NET etc. The fundamental components of web service technology are as follows (see Fig.1) [2]:

1. ***Universal Description, Discovery and Integration (UDDI):*** This framework determines the location and definition of the web services. It can be considered as a successor of Uniform Resource Locator (URL) used on WWW.

2. ***Web Services Description Language (WSDL):*** This component is used to describe the various services that operate on XML messages pointed to UDDI. It is considered as a successor of HyperText Markup Language(HTML) on WWW.

3. ***Simple Object Access Protocol (SOAP):*** This component is an XML based protocol for exchanging information in distributed environment. It is considered as a successor of HyperText Transport Protocol (HTTP) on WWW.

3 International Electrotechnical Commission (IEC) Standards

The activities of the Technical Committee 57 (TC 57) of the International Electrotechnical Commission (IEC) are concerned about the development of the standards for power systems management and associated information exchange. This work has been spilt into various working groups. Two among them are developing standards for application integration in utility control center. Their details are as follows:

1. **Working Group 13 (WG 13):** This is developing IEC 61970 standard for the Energy Management Systems (EMS)/ Supervisory- Control for- Data Acquisition (SCADA) [5].
2. **Working Group 14 (WG 14):** This is involved in developing IEC 61968 standard for distributed systems [6].
3. **Working Group 10 (WG 10):**The IEC 61850 standard is for designing a Substation Automation System (SAS) [4].

3.1 Need for IEC Standardized Web Services

The International Electrotechnical Commission has proposed a certain set of standards for the web services for the application integration in the power utility systems. The main objective behind the introduction of these standards is that the normal web services are lacking in a few aspects:

1. **Point-to-Point integration:** An application that needs to interact with a web service in an external application will need to establish independent links with each external application. It may work fine if it involves one or two applications. But for n number of applications it needs to establish n(n-1) integration paths. Thus it fails to provide point-to-point integration for even moderately sized applications.
2. **Lack of Standardized Data Model or Service Definitions:** The World Wide Web Consortium (W3C) does not offer any definition for standardizing the data exchanged between applications. If two vendors utilize a web service interface, one vendor will provide a different set of services using a different data model compared to a similar system from another vendor. Due to this, we need to customize the application integration for a specific set of applications. This results in a high dependence on the customization for such off the shelf applications.

The issues related to the web services can be solved with the help of GID and CIM.

1. **Point-to-Point integration:** To eliminate point-to-point integration issue, the standardized web services make use of Service Oriented Architecture (SOA) using Enterprise Service Bus (ESB). Enterprise Service Bus is an application framework which takes the applications and divides them into individual business functions and processes which are referred to as services. The SOA wraps all business functions from the new and existing applications with an interface using W3C standards. The main function of ESB is to combine SOA with message routing, transforming messages using publish/subscribe to provide a single point of integration. Due to this, an application gets a single view of all the applications thereby eliminating the need for establishing separate links with each external application.
2. **Lack of Standardized Data Model or Service Definitions:** To eliminate the issue of lack of standardized data model or service definitions, the standardized web services make use of Generic Data Definitions (GID) for standardized definitions and Common Information Model (CIM) for standardized data models (GID and CIM are dealt with in the sections 4.1 and 4.2 respectively). The web service interfaces which conform to GID and CIM standards can interoperate with one another without programming. The CIM used in applications from two different vendors could support exactly the same data model thereby eliminating the need for customization.

4 IEC 61970

The IEC 61970 standard consists of the Common Information Model (CIM), Common Interface Specification (CIS), a system topology for exchanging of graphical schematic information. The interface services used in IEC 61970 are a part of CIS. These interface services are called as Generic Interface Definition (GID) [2].

4.1 Common Information Model (CIM)

The Common Information Model (CIM) is used as a standardized data model for exchanging the power system data. It is used in the electric domain for distribution and transmission energy management systems. The CIM model is developed using Unified Modeling Language (UML). The need for CIM was evolved due to deregulation and the upcoming needs for the exchange of power data between companies using systems of different vendors connected via web technologies. The aim to develop a common data model for power data representation and an exchange format became more and more evident. In CIM, the objects are represented as classes having attributes and relations to other classes[3], as is in the usual way in UML.

4.2 Generic Interface Definition (GID)

The main goal of Generic Interface Definition is to provide standardized interfaces for the applications. The applications use this definition interface for exchange of information. They also help to map specific technology profiles which provide standardized interfaces supporting interfaces based on C++ programming language and interfaces based on World Wide Web Consortium (W3C) [2].

The GID interfaces defined by IEC 61970 are further categorized into four seperate interfaces:

1. **Generic Data Access (GDA):** This interface is based on the concept of request/reply service that provides access to generic data. The specification of GDA includes services that are helpful in implementing a master resource identifier (MRID), a unique identifier for the construction of any power system resource defined in Common Information Model (CIM). It also includes the namespace generation for avoiding the element name conflicts. It provides the services to access and manipulate an object oriented server using the CIM and a method by which the CIM model server can issue notices to the client applications. The working of GDA is based on the work of other standard organizations including the Object Managements Groups Data Access Facility (DAF) [7].
2. **High Speed Data Access (HSDA):** This interface is designed to enable the systems to exchange real-time data. It allows the clients to browse through the servers for exchange of data as well as ask for updates to the data-points using namespace derived from Common Information Model (CIM). The HSDA interface can be used to interoperate with the other IEC interfaces. The HSDA is specialized to real time applications. It also supports subscription and read/write operations. It also helps to write new values to the server. This interface is platform independent and is based on the work of the other standard organizations like the OPC Foundation Data Access (DA) [8].
3. **Time Series Data Access (TSDA):** This interface is designed to enable the systems to exchange historical (time series) data. It allows the clients to browse through the servers and query for the data which existed at a specific point of time or over a range of time using the namespace derived from the Common Information Model (CIM). This interface also supports subscription and read/write operations. The TSDA is specialized for the historical data applications. The TSDA interface is platform independent and is based on the working of other standard organizations like OPC Foundations Data Historical Access (HDA)[9].
4. **Generic Eventing and Subscription (GES):** This interface is designed to enable the publishing and subscribing of generic XML based messages and also the messages specified under IEC

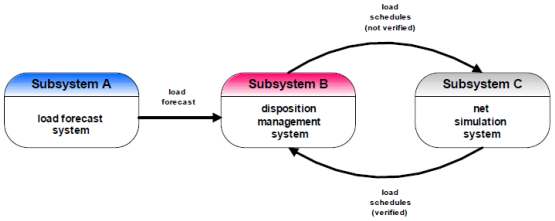


Fig. 2. Communication in EMS [3]

61970. GES helps the subscribing application to browse the messages that a publishing application supports using the namespace derived from the Common Information Model (CIM). It deals with generic data exchange as compared to HSDA and TSDA, which deal with specialized applications [10].

Implementation of CIM along with GID serves the following benefits [2]:

1. **Off-the-Shelf Interoperability of Applications:** GID, in combination with the CIM would help in the integration of two independently developed applications without extensive programming and customization.

2. **Enablement of Third Party Market:** The interfaces and data models in proprietary format is unique to the specific application supplier. This reduces the possibility of working of the third party applications. With GID, the interfaces and data models are standardized thereby allowing the independent development of applications. The Third party market will benefit from the interoperability of products resulting in an increase in product availability and competition.

3. **Reduced Configuration Costs:** The GID interfaces provide the information in the context of the CIM. The configuration of applications can be obtained in an automated manner thereby reducing the integration configuration costs.

4. **Improved Usability of Integration Infrastructure:** The client application fails to understand the data representation that each server application use internally. This is due to the fact that data accessed using GID services is represented in the context of the CIM. The data that is placed in the integration infrastructure using CIM will not only lower the training costs but also helps in improving and understanding of the data exchange in an integrated system.

5. **Future Flexibility:** Due to portability of GID services, they can be based on widely used technologies such as SOA, Enterprise Service Bus (ESB) and the web services. In the future, we expect the same GID services can be ported into the new technologies thereby making the transition simple and easy.

5 Application of CIM in EMS

An Energy Management System is composed of a set of different subsystems and services building the whole functionality of the entire system. Although the subsystems may be produced from different vendors working on different system platforms and data exchange formats, there is always a need to exchange data between subsystems [3]. For example, a substation A calculates the optimized load schedules for a power grid. It provides the input data for a subsystem B which calculates corresponding load estimate for distributed power grid. There is a subsystem C which verifies the load schedules using simulation and again sends the result to subsystem B.

This communication can be shown as in Fig.2 The following are some of the benefits of CIM in EMS:

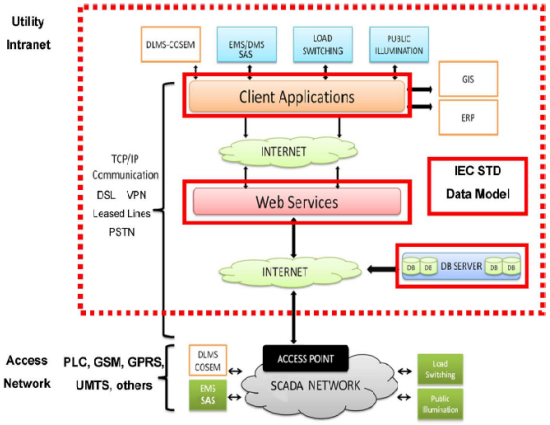


Fig. 3. Implementation based on Websrvices and IEC Standards [1]

1. **Support of a Service-Oriented Architecture (SOA) by using a common language:** A communication and integration platform can be used to provide communication and data exchange between the different systems in a heterogenous environment. For efficient communication it is necessary to build adaptors for the transformation to get data interpretation. The building of adaptors is very expensive. The CIM provides various options to support communication between systems in a heterogenous environment. The systems connected to EMS provide internal data based on CIM and provide external CIM-compliant interfaces. Due to this, there is less data conversion routines within adaptors and systems can work on same data models. A service-oriented architecture supports CIM standard to provide an integration platform for various subsystems.

2. **Improvement in commercialisation:** The commercialisation of a system depends on the functionality as well as flexibility to integrate with other systems. This helps in the reuse of existing infrastructure and also improves the overall performance of the system. The scope of commercial products in the market depends on the standards set by themselves.

6 A case study: The role of web services in controlling and Monitoring services for EMS/SCADA

In the following section and thereafter, we shall present a case study of the role of web servies in controlling and monitoring services for EMS/SCADA . The implementation is open source based which aims at decoupling applications, reducing upgrading costs and facilitating easy procurement process of the new applications [1]. In the case study, the choice of information exchange has been based on a IEC standardized data model using the webservices. The model aims to maximize network centric operations which are summarized in Fig.3

6.1 Methodology to support implementation testing

The implementation testing is basically used to validate the web server that is being developed. The methodology which supports implementation testing is the database design. The IEC services for EMS/SCADA systems are basically related to monitoring and controlling activities of physical devices. In the database design, physical devices are represented by database tables and their characteristics are represented by database table attributes. The Entity-Relationship (ER) is used in database design for representing physical devices like meters, generators, breakers etc. The main aim is to simulate the reading actions on physical devices on IEC-compliant databases.The data model designed is based on the HSDA interface and TSDA interface, which are disccused in Section 4.2. The CIM is used for object-oriented representation for the data model. As the CIM is based on Unified Modeling Language (UML) notation, it is used to define the

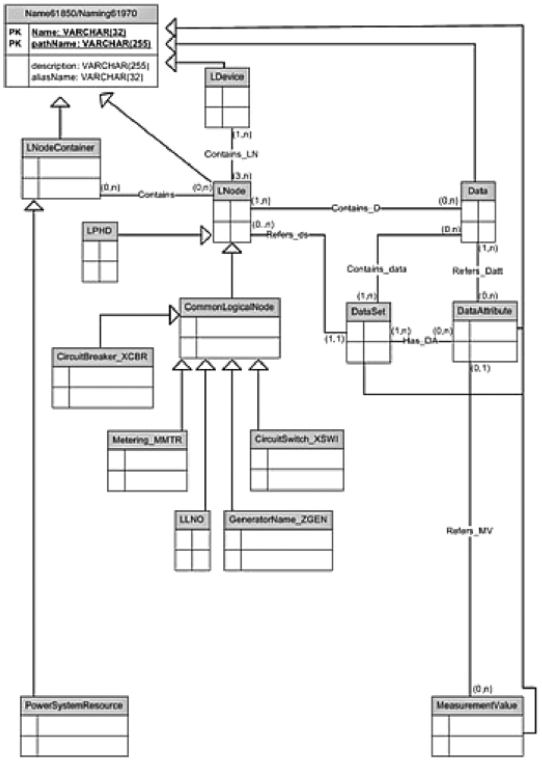


Fig. 4. New entities defined for IEC 61850 substation data model representation [1]

name for each class, its attributes and relationships with other classes. This scenario deals with the translation of CIM UML notation into ER notation. The HDSA interface provides direct access to the devices represented by database tables. It also provides the direct interaction with the simulated device in terms of reading and setting controllable parameters. The TSDA interface described in IEC 61970-407[11] standard is used for reading and writing time-series data. The TSDA database design is similar to HSDA design. The main objective is that if the value in the physical device attribute stored in the HSDA database table is updated, this updated value is stored in time-series database. The important function of TSDA database manager is to keep a record of historical data and also uses primary key of the system as a code to represent the identifier of the insertion.

6.2 Integration and Extension of IEC Standards

In this section, we discuss about the integration of IEC 61970 and IEC 61850 data models. The IEC 61850 is a standard for the design of Substation Automation System (SAS). For a detailed exposition, the reader is referred to [12],[13],[14], we omit these details due to the space limitations. Substation Automation System (SAS) defines the communication between devices in the substation and the related system requirements. In order to integrate the data models of IEC 61970 and IEC 61850, we generate an extension for SAS data models (IEC 61850) and IEC 61970 data model and CIM. The Electric Power Research Institute(EPRI) provided a basic extension for the integration of both the data models. The articulated extension involves merging of IEC 61850 -7-3 [13] and IEC 61850-7-4[14] into IEC 61970. The IEC 61850-7-3 standard describes common data classes and the IEC 61850-7-4 standard describes the logical node classes. Fig.4 shows that the new entities have been defined to represent IEC 61850 substation information into IEC 61970. The new entities defined in the Fig.4 represent the logical nodes of IEC 61850-7-4 standard. These entities are exploited to acquire the specific information and characteristics related to SAS devices. The new entities displayed in the Fig.4 includes

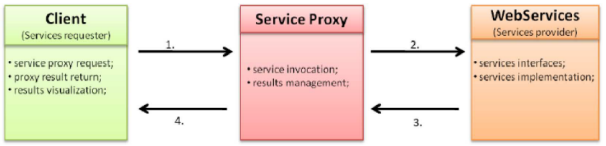


Fig. 5. Interaction between webservices and service proxy [1]

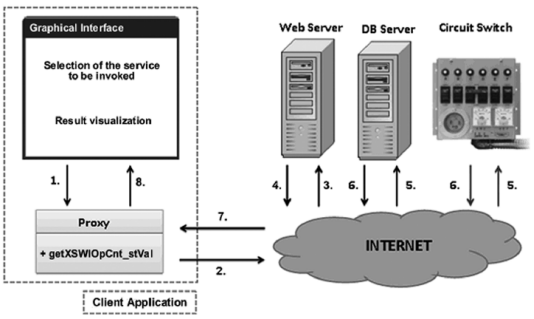


Fig. 6. Interaction Scenario [1]

circuit breaker, metering, generator and circuit switch. As discussed earlier, the IEC 61850-7-4 standard presents the logical node classes and attributes represented as common data classes (IEC 61850-7-3) but not as simple types. We need to define them as simple types for designing the database models. This can be done by merging the information obtained by IEC 61850-7-4 and IEC 61850-7-3 standards. This can be done by mapping with simple data types of IEC 61850-7-3 is provided for the attributes of the logical nodes related to a specific common data class (IEC 61850-7-4). The new attributes have been defined by a composition of logical node attributes and common data classes. For example, the attributes defined in circuit- breaker entity are Loc_StVal, Loc_Q and Loc_TS. The Loc is the name of the arribute of the logical node specified by the IEC 61850-7-4 standard. This attribute also represents the common data class single-point status(SPS) provided by IEC 61850-7-3 standard. The attributes defined in the SPS include status value StVal, quality Q and time-stamp(TS). Loc attribute of the circuit-breaker logical node is composed with StVAl, Q and TS. Therefore, a new set of attributes like Loc_StVal, Loc_Q and Loc_TS are obtained.

6.3 Implementation of Standardized Web Services

The implementation of standardized web services is realized by Java Apache Axis2 web services [2]. The monitoring and controlling services like EMS/SCADA use HTTP as the transport layer. The IEC standard interface requires the implementation of asynchronous and synchronous calls as well as event handling. Java Axis2 is not capable of providing an event module. The exchange of structured information is done by Simple Object Access Protocol (SOAP). SOAP is an XML- based protocol which consists of different parts and is the foundation of web services stack layer. The implementation pattern used for monitoring and controlling services for EMS/SCADA is called as Service proxy pattern. This integration pattern is a part of Domain patterns. The main objective of this pattern is to perform application decoupling. The interaction between parts is shown in Fig.5. This figure represents three proxies which contain Java methods. These methods represent services provided by the web services. In order to access IEC-compliant web- services we need to include IEC-compliant proxy server. The main function of proxy modules is to provide data formatting when the invocation of service returns different data types. The advantages of using proxy servers are there is no direct interaction between client and service.

It also permits reusing of implemented services in other parts of application. The idea behind using a service proxy can be explained

with the help of the following Fig.6. This figure shows the interaction between Graphical User Interface (GUI), web services and physical devices. The steps involved in this process are as follows:- The GUI/client sends a request for a specific service. This request is managed by the service proxy which is local to the client application. The service proxy invokes the web services. The service implementation at web services provides the reading of the database table which contains the required information. The last step is to come back to the GUI/client which provides result visualization in a real implementation. The realized server provides the writing and reading of typical values stored in database tables which represent physical devices. The services offered by Axis2 includes how to get and set parameters of physical devices and also to read time series data stored in a database. Timestamp management is also take into account for the conversion between java.sql.Date and java.util.Date[2]. This is because the database treats the date as java.sql.Date whereas web services treat it as java.util.Date.The implementation phase involves implementation of several Java and XML files. The Java files are relative to proxies and services implementation. XML files are implemented to configure Axis2 web services for correct compiling of source code, classpath settings and execution instructions. It also used to specify message receivers and an operation list. Web Services Description Language (WSDL) in XML format is used to describe the network services as a set of endpoints on messages in document- oriented or procedure oriented information.

7 Conclusions

In this paper, we have discussed the role of the standardized web services in the integration of Electric Utility Control Center applications. The normal web services suffer from disadvantages like point-to-point integration and lack of standardized data models, which led to the development of the standardized web services. The International Electrotechnical Commission is responsible for the development of these standardized web services. Various standards have been proposed among them IEC 61970, IEC 61850 focus mainly on the applications of electric utility control center. These standards help in overcoming the demerits of the normal web services. They also help in the successful integration of the business utility applications with the power system operation applications, which is quite a challenging task. There are certain benefits which result in combining CIM and GID components of IEC 61970 standard in Energy Management Systems. These are explained in detail in the paper, which substantiates the role of the standardized web services in the electric utility control center applications integration. Further the case study about the open source implementation of web services as well as integration and extension of IEC 61970 and IEC 61850 standards is discussed in the paper. These are investigated in the context of controlling and monitoring services for the EMS/SCADA system. We find that the standardized web services are suitable for dynamic working environments. It is also observed that the timelines are reduced dynamically by building the integration infrastructure for operational applications. Further, the cost, complexity and programming effort to build and maintain the integration infrastructure is also effectively reduced using these standardized web services.

References

[1] Andrea Mercurio, Alessandro Di Giorgio, Pierfrancesco Cioci, “Open-Source Implementation of Monitoring and Controlling Services for EMS/SCADA Systems by Means of Web Services IEC 61850 and IEC 61970 Standards”, 1148-1153, Volume 24, 2009,IEEE transactions on power delivery.

[2] R.E.Mackiewicz, “The Benefits of Standardized Web Services Based on the IEC 61970 Generic Interface Definition for Electric Utility Control Center Application Integration”, 491 - 494,Oct. 29 2006-Nov. 1 2006 , Power Systems Conference and Exposition, 2006. PSCE ’06. 2006 IEEE PES.

[3] Mathias Uslar, Tanja Schmedes, Andreas Lucks, Till Luhmann, Ludger Winkels, Hans-Jurgen Appelrath, “Interaction of EMS related systems by using the CIM Standard”, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.65.1546.

[4] IEC TC57 - WG10/17/18, http://tc57wg10.info.

[5] IEC 61970 - Energy management system application program interface (EMS-API); International Electrotechnical Commission, Geneva Switzerland, http://www.iec.ch/

[6] IEC 61968- Application integration at electric utilities - System interfaces for distribution management; International Electrotechnical Commission, Geneva Switzerland, http://www.iec.ch/

[7] Object Management Group, Utility Management Systems, Data Access Facility (DAF), June 2005, http://www.omg.org/spec/

[8] OPC Foundation, OLE For Process Control Data Access Specifications, June 2002, http://www.omg.org/spec/

[9] OPC Foundation, OLE For Process Control Historical Data Access Specifications, January 2001, http://www.omg.org/spec/

[10] OPC Foundation, OLE For Process Control Alarms & Events Specifications, October 2002, http://www.omg.org/spec/

[11] Energy Management System Application Program Interface (EMS-API), Part 407: Time Series Data Access (TSDA), 2007 IEC 61970-407, Int. Electrotech. Comm. [Online]. Available: http://webstore.iec.ch/webstore/webstore.nsf/artnum/038213?opendocument.

[12] Communication Networks and Systems in Substations, Part 7-2: Basic Communication Structure for Substation and Feeder Equipment, Abstract Communication Service Interface (ACSI), 2003 IEC 61850-7-2, Int. Electrotech. Comm. [Online]. Available: http://webstore.iec.ch/webstore/webstore.nsf/artnum/030581?opendocument.

[13] Communication Networks and Systems in Substations, Part 7-3: Basic Communication Structure for Substation and Feeder Equipment, Common Data Classes, 2003 IEC 61850-7-3, Int. Electrotech. Comm. [Online]. Available: http://webstore.iec.ch/webstore/webstore.nsf/artnum/030583?opendocument

[14] Communication Networks and Systems in Substations, Part 7-4: Basic Communication Structure for Substation and Feeder Equipment, Compatible Logical Node Classes and Data Classes, 2003 IEC 61850-7-4., Int. Electrotech. Comm. [Online]. Available: http://webstore.iec.ch/webstore/webstore.nsf/artnum/030606?opendocument.

Online Voting: Yes or No

Klaas Mussche, and Edwin-Jan Harmsma

Abstract— We introduce a new design of an online voting system, based upon a concept called *trusted persons* and the usage of *public-key cryptography*. Online banking systems and government services via the Internet gain popularity, but voting is not often done via the Internet. We think that this is because people tend to distrust online election systems, since they have to guarantee properties like *fairness*, *transparency* and *anonymity*.
In our research we look at the requirements for fair elections and translate these to technical and social requirements for online voting systems. We identify the general problems online voting systems have to deal with. We present a system that meets these requirements and addresses potential problems. Finally we compare our system with two already existing systems, the Rijnland Internet Election System (RIES) used in 2004 for the elections of the Dutch local authorities for water management, and a secure anonymous Internet voting system designed by Yu-Yi Chen, Jinn-Ke Jan and Chin-Ling Chen.
This paper focuses on online general elections, like for the House of Representatives of the Netherlands (Tweede Kamer). We assume that all votes are handled by the online voting system and that for example voting with paper ballots is not allowed.

Index Terms—Cryptography, Elections, Online voting, Online democracy.

1 Introduction

Nowadays more and more tasks are performed online. A lot of people are currently using online banking systems and some government related tasks can also be performed online. However, elections are still rarely performed via the Internet. In this paper we describe how electronic voting systems could be used for national elections, e.g. for the House of Representatives of the Netherlands (Tweede Kamer). The main research question in this paper is: *What are the requirements, possibilities and problems for online national elections?*

First, we show an overview of the technical requirements for such a system, and most important, we explain the difficulties of these requirements. The interesting part of this is that some requirements like transparency and anonymity might conflict at first sight. However, some of these issues can be solved by using public-key cryptography in a smart way.

Secondly, with these requirements in mind we introduce and describe the design of a new online voting system. This system opens a new perspective for generating the private keys of the voters, this task can be performed by both the voter itself or by the government party. While in existing systems this is always performed by a selected party or the government itself. Another important aspect is that our system is able to distinguish votes that are submitted in a regulated voting place and votes that are submitted via the Internet. This is done by having a specific *trusted person* for these regulated votes. Finally, the user is able to select his own *trusted person* for voting via the Internet, this makes the system more robust since no single party is responsible for the *fairness* of the system. We assume that all votes are handled by the online voting system, and that voting with paper ballots is not allowed.

During our research we studied two different online voting systems; The *Rijnland Internet Election System* used in 2004 for the elections of the Dutch local authorities for water management and designed by Engelbert Hubbers, Bart Jacobs and Wolter Pieters [5] and the proposed system in *The design of a secure anonymous Internet voting system* by Yu-Yi Chen, Jinn-Ke Jan and Chin-Ling Chen [3]. Finally, we discuss the suitability for national elections of these two existing voting systems, and compare these systems with our proposed system. A detailed comparison with respect to the discussed requirements is shown in the end.

- *Klaas Mussche is master student at the University of Groningen, E-mail: klaasmussche@gmail.com.*
- *Edwin-Jan Harmsma is master student at the University of Groningen, E-mail: ejharmsma@gmail.com.*

2 Problem statement

An online voting system must guarantee several important requirements to become accepted in a democracy. Requirements can be technical, for example to prevent voting more than once, but also be more social related, for example if people do understand the voting process. In this paper we will mainly focus on the technical requirements, however, we will first briefly discuss some important social requirements.

2.1 Social requirements

Social requirements do not relate directly to the design and implementation of the system. However, these requirements will influence the design of the system in a more general way.

The requirements listed below are in more detail described by Stephen Coleman in [4].

Turnout A high percentage of the eligible voters should participate in an election to have a good reflection of the public's will.

At this point online voting could play an important role, since it in general becomes easier to vote if people can do it at home or work.

Representativeness The final outcome of the election should be a good reflection of the socio-demographic composition of the eligible electorate. This might be in contrast with the *Turnout* requirement, since a higher turnout does not directly improve the representativeness.

Especially online voting might affect the representativeness of an election in a negative way because the participation of a group that is more familiar with computers might increase, while lower educated or elderly people might get under-represented.

Transparency People must understand the democratic system, and how the voting procedure is fulfilled. Voters should be able to understand that their vote is handled secure and anonymous. Furthermore, the overall process of voting (e.g. the process of counting the votes) should be clear for a nontechnical voter.

This social requirement might become a real problem, since the majority has no clue about the mathematical details that are required to understand for example the encryption that ensures the privacy. On the other hand, the mass society currently uses online banking systems without having any knowledge about technical details.

An online voting system is only transparent if the voters can ensure that no one else knows their private key. Each voter might

need a private key. If this private key is generated by the government or another third-party organization, then voters cannot be sure that this organization has really deleted their private keys and thus cannot be sure no one is able to vote in their name. Therefore, a transparent voting system requires or at least allows voters to generate their own private key.

2.2 Technical Requirements

We used the requirements from Yu-Yi Chen et al. [3] as starting point for our research. The requirements listed in this paper are not specifically designed for national voting systems. However, the basic principles described in this paper are suitable for national elections. To make the requirements more suitable we modified some of them as is shown below.

Fairness No one can learn the outcome before the tally.

Eligibility Only permitted voters are allowed to submit a vote that contributes to the outcome of the election.

Equality Every voter contributes equally to the final outcome.

Uniqueness Every voter can only contribute once to the final outcome. But a voter should be able to submit his vote multiple times to prevent family voting.

Yu-Yi Chen et al. describe in [3] that a voter must not be able to vote more than once. We modified this requirement so the voter can vote more than once, but only the last submitted vote contributes to the final outcome. This preserves family voting because a voter can now resubmit a ballot until a certain deadline. Family voting is in this case only prevented if the bribed or suppressed voter is able to go to another (public) computer before the deadline. If the voter is in a suppressed environment short before the deadline then family voting is still possible, but this is solved by the *Freedom* requirement.

Freedom A voter should be able to submit his ballot in a regulated place. In practice this means that a ballot which is submitted in a trusted polling place should overrule all previous and potential upcoming submissions.

In this way it is guaranteed that a voter is able to perform his vote without being influenced by others.

Uncoercability A voter can not prove (afterwards) how he voted. This is highly related to the *Freedom* requirement and is required to prevent bribery.

Anonymity There is no way to derive a link between the voter's identity and the marked ballot.

Accuracy A voter's vote can not be altered, duplicated, or removed.

Efficiency The computations can be performed within a reasonable amount of time.

Robustness A malicious voter cannot frustrate or disturb the election.

Mobility A voter should not be restricted in the place where he or she submits his ballot. This is also related to the *Freedom* aspect, because the voter should also be able to vote in a regulated place.

Practicability No extra skills should be required to vote, but some additional equipment might be required (e.g. equipment to identify the voter).

This is required to allow every person that is part of the democracy to participate in the elections. (*Representativeness*)

3 Design

Traditionally, people who are eligible to vote can cast their vote for a certain candidate on the day of the election in a voting station. Voting is then done by filling in a box on a voting ballot. Those ballots are collected in ballot boxes. After the time to vote has passed, people will count how many votes each candidate has. This is called the tally. Formal procedures are used to ensure that no one can vote more than once and that no one can influence the results or count the already collected votes before the tally. It usually is possible for voters to let another voter vote in their name as well.

This scheme more or less stays the same when elections are held via the Internet. The traditional paper ballot will be replaced with a digital version. Physical boxes are no longer necessary to gather the ballots. Time consuming manual counting of the votes can be avoided: instead, computers can do the tally, probably faster and more accurate than humans can ever do. However, some of the formal procedures to guarantee fair elections cannot be applied digitally, and have to be replaced, for example with a cryptographic solution.

3.1 Most important requirements

Probably the most important requirement for an online voting system is transparency. When we assume that people do trust the traditional voting system, than it is a good choice to design our online voting system in such a way that it resembles the traditional system. For example, with traditional voting voters can verify that their name is not on the ballot, so they know that voting is anonymous. The formal procedures at the voting center ensure that nobody can vote more than once, and the counting of the votes is done by other humans. Every online voting system will have the drawback that laymen cannot verify for themself that the system guarantees fair elections. But if the online voting system resembles the traditional system, then laymen might trust the system. Professionals with enough technological knowledge should be able to verify the correctness of the system for themself.

Online voting must feel inherently safe. To obtain this feeling, it is necessary to replace the formal procedures which cannot be applied via the Internet with technical procedures which work via the Internet. Most people might not be able to understand the technical procedures in detail, and even the tiniest uncertainty about the validity of the system can scare lots of people. To let the system feel safe, it should not be designed as a set of computer programs, but as a set of protocols and algorithms. Computing scientists can verify that those do not leave any possibility for abuse. That way no company, organization or individual can have such a role in the process, that voters can think that they might be able to influence the results.

Laymen probably trust other people better than machines and algorithms, because they do not understand the latter at all. In our system we introduce the concept of *trusted persons*. In the traditional scheme, people have to trust the voting center staff. This staff secures the votes until the tally, and ensures that all votes will contribute to the final outcome. Using our trusted persons approach, voters can choose for themself who they want to trust with securing their digital vote. However, these trusted persons should not have the ability to influence the final outcome.

3.2 Technical overview

Security keys

Each voter has to get a public and private key prior to the election day itself. An online voting system needs a cryptographic key scheme for encryption, decryption and signing of the votes. Traditionally, people who are eligible to vote get a voting pass by name, so the voting center staff can verify that the person is eligible to vote. The voting center will keep the voting pass, so people cannot vote more than once. We replaced the traditional voting pass with a public and private key for each voter. Some other schemes like [3] and [5] do have a central authentication center which creates the key pair for each person eligible to vote. However, voters do have to trust that the authentication center will not abuse the private keys to influence the election.

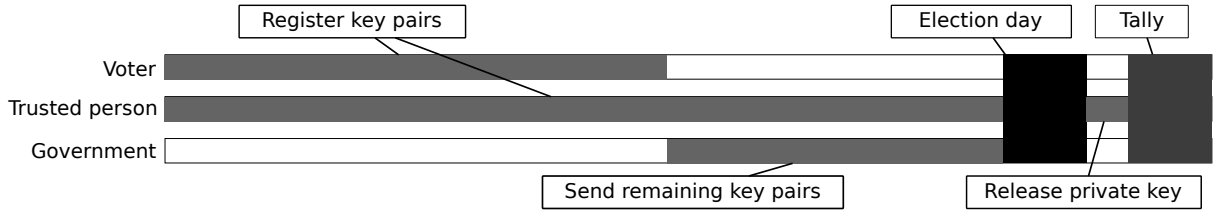


Fig. 1. Timeline of the election process.

Trusted persons do have another public and private key pair. The public key of this pair is published before or at the start of the election day, and voters have to use this public key to encrypt their ballot. At the end of the election day, all trusted persons publish their private key as well, therefore allowing everyone to decrypt all ballots.

Voting

The actual voting is done by a voter by creating a *valid* digital ballot. To create a valid digital ballot, the private key of the voter is necessary, as well as the public key of one trusted person. The resulting digital ballot is encrypted, which means that it is impossible to find out which candidate the voter supported, unless the private key of the trusted person is known. Digital ballots do not contain any link to the original voter, but for each pair of digital ballots it is possible to verify that these ballots where created using different voter private keys.

Figure 1 shows a timeline of the voting process. Initially, all voters do have the time to create and register their own key pairs. After the time for voters to create their own key pairs has passed, the government will create key pairs for those who did not do it themself, and sent these keys to the voters. The government in principle does have the time to do this until the election day. Trusted persons can create and register key pairs also from the beginning and till the election day. At the day of election, all voters can cast their vote. After the time to vote has passed, a short time is given to the trusted persons, so they can release their private keys. When that is done, everyone is able to do the tally by downloading all data and counting the valid votes.

Figure 2 shows another overview of the voting process. It basically shows that there are four keys involved in each vote: the public and private key of the voter, and the public and private key of the chosen trusted person. The voter public key can be used to verify that the voter was eligible to vote. The keys are used to create a digital ballot, which is then stored in the digital ballot box.

Digital ballot box

All digital ballots are gathered into a digital ballot box. The digital ballot box should fulfill the following properties:

1. Everyone should be able to submit digital ballots to the digital ballot box via the Internet without authentication.

2. Voters should have the possibility to verify that their vote was received and stored correctly by the ballot box.

The digital ballot box is a very crucial part of the voting system, because no one can vote if there is no digital ballot box available. For example, a DDoS attack on the digital ballot box can seriously disrupt the elections. However, we think that the digital ballot box does not have to be a weak part of the system. All digital ballots are encrypted on the machine used to create and submit the ballot to the box. No one can decrypt the ballots without knowing the private keys of the trusted persons, so the content of the digital ballot box only give information about how much people have voted, but no preliminary election results. So if one is able to break the digital ballot box, no secret information can leak to the public. However, all digital ballots are sent to the digital ballot box via the web proxies, so the digital ballot box may be on a private IP address so that a DDoS attack becomes more unlikely. In addition it is possible to use multiple redundant servers and (commercially) available DDoS defense solutions to further secure the digital ballot box.

Vote counting

Everyone can count the votes for each candidate when the election is done. All ballots are published after the election, but no one is able to determine who created which ballot except for their own.

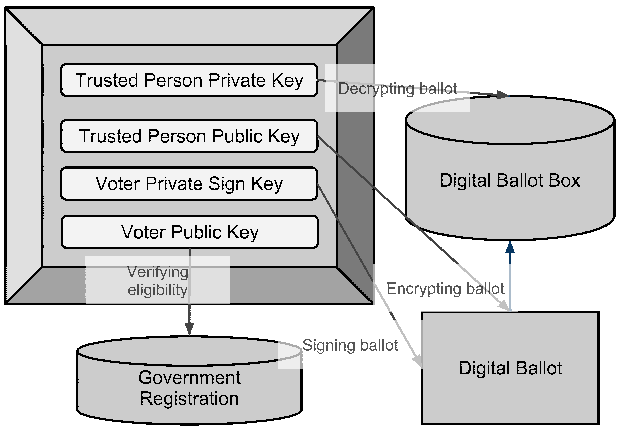


Fig. 2. Overview of the online voting system

3.3 Technical details

The digital ballots are created by the voter, but the voter should not be able to choose the timestamp of the ballot. The web proxies are controlled servers, so they can be used to create the timestamp. However, to give the voter the idea that he or she can verify the timestamp, the user should explicitly accept the timestamp as added by the web proxy to the ballot. So, the voter first creates the digital ballot, but without an timestamp. This ballot is then send to a web proxy, which adds a timestamp to the ballot and remembers that it had added that timestamp to the ballot. The ballot including the timestamp is then send back to the voter, who can verify the timestamp and sign the whole ballot again, thereby declaring that the timestamp is correct. The signed and timestamped ballot is then again send to the web proxy, which verifies if the user did not change the timestamp and then finally forwards the ballot to the digital ballot box.

This extra step of signing the ballot again might look somewhat tedious for people. Also, in controlled voting stations this is not really necessary. So governments might choose to give voters the option to immediately say that they do trust the attached timestamp. In that case, some information is added to the plaintext ballot indicating that no explicit timestamp signing is necessary. However, in that case the timestamp on the ballot can be changed after it has been put into the digital ballot box without any way to detect it, because the combination of ballot and timestamp has never been signed. This can be solved by letting the voter add a *automatically accept* timestamp range to the plaintext ballot. If the web proxy sees that the timestamp for the ballot is within that range, then no explicit re-signing by the voter is

necessary. This timestamp range should not span more then a certain maximum amount of time.

The *life* of a digital ballot is as follows (see also Figure 3):

- The *plaintext* ballot is created by the voter, and contains only the candidate identifier the voter wants to support.
- The *encrypted* ballot is obtained by encrypting the *plaintext* ballot with the trusted persons public key.
- The *signed* ballot consists of the *encrypted* ballot combined with the unencrypted public keys of the voter and the trusted person, and signed with the voter private key.
- The voter sends the *signed* ballot to a web proxy. This web proxy ensures that information like IP addresses are not stored together with the ballot, and will create the *time-stamped* ballot by adding a time stamp to the *signed ballot*.
- The web proxy returns the *time-stamped* ballot to the voter, who verifies the time stamp, and again signs the ballot. The resulting ballot is the *time-stamped signed* ballot.
- The voter sends the *time-stamped signed* ballot back to the web proxy.
- The web proxy forwards the *time-stamped signed* ballot to the digital ballot box.
- At the tally, the public keys of the voters are used to invalidate all votes but the last of voters who voted more than once.
- The public keys of the trusted persons are used to select to private key of the trusted person for each ballot.
- The *plaintext* ballot is again obtained by decrypting the ballot.

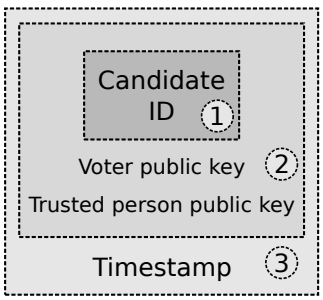


Fig. 3. Digital ballot structure. 1: Encrypted with trusted person public key. 2: Signed with voter private key. 3: Again signed with voter private key.

Each voter and each trusted person does have a public and private key. Trusted persons probably are voters as well, and thus they will have two public and two private keys. These key pairs cannot be the same, each key pair has to be unique. Those keys are used in the following situations:

1. Trusted person public key is used to encrypt the plaintext ballot.
2. Voter private key is used to sign the encrypted ballot twice.
3. Voter public key is used to verify the sign of the encrypted ballot.
4. Trusted person private key is used to decrypt the plaintext ballot.

Submission of digital ballots is done via a web proxy and over a SSL connection. The web proxy is responsible for removal of all information which might comprise the anonimity of the voter, like sender IP address, and attaches a time stamp to each vote. The SSL connection ensures that man-in-the-middle attacks or eavesdropping is not possible.

Our system does not really solve the threat of man-in-the-browser attacks. People who vote should use their own computer, and thus are themself responsible for a secure system. People who don't have access to a trusted computer should cast their vote in the regulated voting center.

Implementation of the overruling behaviour of votes cast at an official voting center is easy possible with the trusted persons approach. Each voting center is another trusted person, and by voting at a voting center the voter implicitly chooses to trust the persons at the voting center instead of someone else. This requires that the public key of voting centers will only be published after the election, because otherwise anyone can just use a voting center public key to create an 'overruling' ballot.

3.4 Practical issues

A correct tally is impossible if one or more of the trusted persons do not publish their private key. This might happen due to illness or accidents. A solution might be to use trusted *groups* instead of trusted persons. Every member of the group knows the private key, and thus can publish the key. However, trusted persons or groups might happen to be not really trustworthy: the whole group might have the goal to not publish the private key and thus disrupt the election. It is probably not possible to prevent this, just ignoring all votes encrypted with this private key is the only option. Voters have to choose which trusted person they do really trust.

People are not good in remembering large and random sequences of text, like the public and private keys. When people create their own key pair, they can store the private key on their computer. The public key has to be registered at the government, but this can be done digital (using for example DigiD for authentication).

4 Comparison

In this study we have done research about existing voting systems as well. We compared the systems described in [3] and [5] with our designed system. We reflected how these online voting systems impact the requirements introduced in Section 2.1 and Section 2.2. The results of this comparison are shown in Table 1.

4.1 The *Chen, Jan and Chen* Voting System

The voting system introduced by Yu-Yi Chen, Jinn-Ke Jan and Chin-Ling Chen in [3] is an important base for the design of our system. We share the idea of most of the requirements listed in this article. However, as with RIES, all the certificates are generated by one single party, which has in our opinion a too important role in the election process.

The system designed by Chen, Jan and Chen prevents bribery since it not possible to verify the voting outcome. If they would allow voters to verify the voting outcome it will be possible for each voter to prove what they have voted. To keep this process transparent for the public, they have chosen to introduce a "supervisor center" that verifies the processes that are performed by the "tally center".

Also is the fairness of the system guaranteed by the fact that the "supervisor center" will monitor if the private keys for decrypting the votes is not released before the voting deadline.

4.2 The *RIES* Voting System

The most important difference of the RIES system is the principle of a "pre-election table". This table is generated before the election starts and contains all possible votes. This table requires that the private DES key of the potential voter is known since the data in this table is created by using this key.

After the creation of the "pre-election table" the DES keys must be distributed among all potential voters. It is essential that the DES

keys will be destroyed after this step, since this is required to keep the voters anonymous. In case that the keys are not properly destroyed the party that generates the keys can create new votes and modify votes afterwards.

Another important aspect of this system is that it allows the user to verify the outcome of the election, and even verify if the submitted vote is actually processed and counted. However, this aspect is of course in conflict with the uncoercability requirement. The voter can now prove what he has voted as long as he saves his key.

4.3 The *Harmsma and Mussche* Voting System

The most important and essential difference between our system and the two previous systems is that the private keys can be generated by the voters itself. In this case, the voter has only to submit his public key to the government, so they can verify the signature. In this process it must be guaranteed that the government will not store the voters personal information attached to the public key. Otherwise, the government would be able to relate the decrypted votes to the voters. If a voter does not want to create his own key-pair, the government is still able to generate the key-pairs for the voters that did not submit a public key in the key-pair registration period. In this case the usability of the system is preserved and suspicious voters are able to keep the important and private information away from the ”evil” government.

Another unique property of our system is that a voter can choose his own trusted person. The trusted person is similar to the ”tally center” in the system of Chen, Jan and Chen. But in our case there can be multiple trusted persons, and in this case the robustness of the fairness requirement is increased.

Related to this final aspect is that the voter is able to go to a regulated place to cast his ballot. The regulated place is in our case just a ”special” trusted person. Votes submitted that are encrypted with a regulated place key have a higher priority than the votes submitted from other places, the regulated vote will overrule all the other votes. In this case, family voting is very unlikely since a voter can still submit a vote in a regulated place which can be done secretly.

5 Discussion

An online voting system requires much more than just a web site where people can cast their vote. This is probably something where people are not aware of. This might cause people to think that the procedures and implementation of online voting systems is unnecessary complex: even if they understand the requirements for an online voting system, they might still be unable to understand how the proposed online voting system guarantees fair elections. However, our trusted persons approach does not only guarantee fair elections, but also gives the voters themselves the control about the fairness of the elections.

Our online voting system does have some disadvantages as well. If people want to cast their vote in a regulated voting station, then they have to carry their public and private key to the voting station. Storage on a digital medium like a memory card is an option, but might be too expensive. Printing it on paper is also an option. Then the voter probably has to type this code over by hand, but he or she might do this wrong. Of course, when people have created their private key on their own computer, and vote online, then key storage is not a problem at all because the key can just be stored on their own computer.

In our scheme the government creates the private keys for *lazy* voters, who didn't do this themself in time. This still gives the government a possibility to influence the results. It might be possible to prevent creation of private keys by the government and key storage problems at all. By adding a key generation facility to the voting stations, voters do have two options: either vote online, which requires the creation of a key pair prior to the election day, or vote in the regulated voting station and create the keys immediately prior to the actual voting. Then they only have to carry a traditional voting pass to the voting center.

6 Conclusion

In Section 3 we presented an online voting system which meets the requirements as stated in Section 2. Due to our trusted persons approach, the ability for voters to create their own private key, and the possibility to vote using your own hardware, we think that our system is *trustworthy*. The idea of trustworthiness for e-government systems is defined by Carter and Bélanger in [2].

Due to Antoniou et al. trust plays *"a mayor role in the way people view and use information systems, lack of trust renders even expensive and sophisticated information systems completely useless"* [1]. Trust is not based on *"some publicly available systematic design process, but rather on the reputation of the system's implementor"* [1]. We did not give the exact implementation of our online voting system, but only the high-level scheme as well as the cryptographic requirements. This means that the design process nor the system's implementor play a mayor role in determining the trustworthiness of our system.

The problems introduced by letting people use their own computer to cast the vote via the Internet introduces a whole new class of design challenges. Volkamer et al. did some more research about this in [8].

Online voting is not new, it has already been used. Both the Netherlands and the UK have used online voting for certain elections. A case study of the online voting discourses in the UK and the Netherlands is performed by Pieters and Van Haren [7]. Estonia recently used online voting as well [6].

We do think that online voting is something which should really be considered. It has several advantages over traditional voting, and therefore is a good alternative for traditional voting. In this article we described the RIES system [5] and the system by Yu-Yi Chen et al. [3]. There are more systems not covered in detail in this article, like the one used in Estonia. We think that our system is a good start if a country wants to start using online voting, however more research in several aspects is required. Some points that might require further research are:

1. The possibility of man-in-the-browser attacks.
2. Do people really use the possibility of creating their own private keys or are almost all voters "lazy".
3. Technical implementation of the various parts.

Acknowledgements

We want to thank dr. F. B. Brokken for doing an expert review of our work.

References

[1] A. Antoniou, C. Korakas, C. Manolopoulos, A. Panagiotaki, D. Sofotassios, P. G. Spirakis, and Y. C. Stamatiou. A trust-centered approach for building e-voting systems. In *EGOV'07*, pages 366–377, 2007.

[2] L. Carter and F. Bélanger. The utilization of e-government services: citizen trust, innovation and acceptance factors. *Information Systems Journal*, 15(1):5–25, 2005.

[3] Y.-Y. Chen, J.-K. Jan, and C.-L. Chen. The design of a secure anonymous internet voting system. *Elsevier*, January 2004.

[4] S. Coleman. Just how risky is online voting? *IOS Press*, 2005.

[5] E. Hubbers, B. Jacobs, and W. Pieters. Ries - internet voting in action. *IEEE Proceedings of the 29th Annual International Computer Software and Applications Conference (COMPSAC'05)*, 2005.

[6] E. Maaten. Towards remote e-voting: Estonian case. In *Electronic Voting in Europe - Technology, Law, Politics and Society*, pages 83–90, 2004.

[7] W. Pieters and R. van Haren. E-voting discourses in the UK and the Netherlands, August 2007.

[8] M. Volkamer, A. Alkassar, A. reza Sadeghi, S. Schulz, and S. Ag. Enabling the Application of Open Systems Like PCs for Online Voting. In *In Proc. of Workshop on Frontiers in Electronic Elections, 2006. [81] Dennis Volpano and*, 2006.

Table 1. Comparison with two other online voting systems

Requirement	Chen, Jan and Chen	RIES	Harmsma and Mussche
Fairness	The vote is stored encrypted at the Tally Center. And can only be encrypted after the voting deadline, this is ensured by ”supervision center”.	SURFnet stores all ”technical votes” which are encrypted and can only be decrypted after the voting by TTPI.	The vote is stored encrypted in the digital ballot box and can only be decrypted with the private keys of the thrusted persons, who will release these keys after the voting process.
Eligibility	It is not possible to vote without a ”personal certificate” that is generated by the ”certificate authority”.	Only voters with a valid DES key, which is generated by TTPI can submit a vote that contributes to the final outcome because the vote will be checked according to the ”pre-election” table.	Voters can create their own key-pairs or wait until the government will create the key-pairs. In both cases a vote is only valid if the public key is registered at the government, in this way it is ensured that only eligible voters can contribute to the final outcome.
Equality	Every vote contributes equally to the final outcome.	Every vote contributes equally to the final outcome.	Every vote contributes equally to the final outcome. Moreover, a voter can submit multiple votes, but only the last one or the one submitted in a regulated voting place contributes to the final outcome.
Uniqueness	Each voter has only one possiblity to get a ”voter-pseudonym signature” from the authority center. Only with this signature it is possible to submit a valid vote.	All duplicate votes are removed, so if a voter submits multiple votes all his votes are invalid and will not contribute to the final outcome.	A timestamp will be attached to every submitted vote, which can be checked by the voter. The signed vote including this timestamp is send to the digital ballot box, this ensures that only the last submitted vote of the voter is counted or if present only the regulated vote.
Freedom	All votes can only be performed with the online system. The user is not able to go to a regulated place to submit an unchangable vote.	The user is not able to vote in a regulated place.	The user can vote in a regulated voting place. In this case all the other votes are overruled by the regulated vote.
Uncoercability	A voter cannot prove what he has voted afterwards.	A voter can prove what he has voted afterwards by performing an election verification.	A voter can prove what he has voted afterwards by performing an election verification.
Anonymity	The private keys of the voters are generated by the ”certificate authority”, this party must remove the private keys after distributing it to the potential voters in order to keep the process anonymous. Also must the process of distribution of the private key be performed secure.	Anonymous elections can only be guaranteed if the secret DES keys will be destroyed by TTPI. The SURFnet party is responsible for removing user critical data like IP adresses and timestamps.	A voter is allowed to create and register his own key-pair. In this case no one else will ever see the private key of the user, the government has only a registration of the public key. The proxy servers will guarantee that no other information is send to the government than necessary.
Accuracy	Both the ”tally center” as ”supervision center” must collaborate to get the correct result. By this separation of duties it is unlikely that votes can be altered, duplicated or removed.	All voters are able to validate their own vote and count the submitted votes.	All voters are able to validate their own vote and count the submitted votes.
Efficiency	The ”tally center” will check for duplicates for each submitted vote. This might make the voting process less efficient if the database of votes becomes large.	During the voting process SURFnet only stores the votes that are send via a HTTPS connection. After the elections all the data has to be decrypted and validated, but efficiency at this part is not a big issue.	During the voting process the digital ballot box only stores the votes that are send via a HTTPS connection. After the elections all the data has to be decrypted and validated, but efficiency at this part is not a big issue.
Robustness	The system might be susceptible for DDoS attacks since the efficiency of the voting submission process can become a problem if the system is flooded with fake votes.	Less susceptible for DDoS attacks since the efficiency of the voting phase is higher.	Less susceptible for DDoS attacks since the efficiency of the voting phase is higher. Also can the digital ballot box be hidden from the public internet, in such a way that only the proxy servers can connect to the digital ballot box.
Mobility	A voter must submit his vote via the Internet.	The real intention of the system is to be only available via the Internet. However, during the elections of the Dutch local authorities of water management it used a hybrid system where people also were able to vote via the regular mail. But the ”outcome check” can only be performed on the online votes, in this case security is not guaranteed for the mail voters.	The voter can submit his vote via the Internet, and also submit his vote in a regulated voting place.
Practicability	The voter must receive his ”personal certificate” in order to be able to vote. No extra equipement is necessary.	A voter must have a secret key to be able to vote. This is done by printing a sixteen characters long key on the ballot. Verifying the election outcome requires additional software and technical knowledge, but this action is not performed by every voter.	The private key must be distributed or created by the user with an application. The actual voting process does not require an application and can be performed by a normal web browser.

Google tools and SEO for efficient web page development

Darius Karremans, Konstantinos Theodorou

Abstract—In this paper we are explaining how Google classifies websites in a search result and which specific tools it provides to improve this classification. Our approach is to explain two specific tools the Webmaster's tool and Google Analytics, useful for most webmaster's, in a way that it is easily understandable by the most people that have a website. These tools can be proven the driving force for a website that wants to increase its visitors and to appear higher in the search engine's results. After analyzing these two tools, we will u1se them in different experiments in order to find the significance of the improvement that is offered.

Index Terms— Google, SEO, search engine, Webmaster's Tools, Google Analytics, website.

1 Introduction

In the internet times that we live, creating a website or a simple web application is less than complicated to do. Content management systems (CMS) and Rapid Application Development tools (RAD), allow people with no particular coding experience to create a good looking website that will serve their purposes. Whatever these purposes are, profit or non-profit, informational or propagandistic, none creates something with having in the back of his/her mind that it is going to fail. Success is the word that surrounds any idea and although translating the word success for a website, we might take several result, we believe that the most important is "visitors".

For a website to be accessible by many visitors it has initially to be easy to be found. Easy to be found is interpreted in our research as the appearance in the results of the first page of a search engine, by using a specific query. We set as a standard the usage of the Google Search Engine, considering that Google is one of the dominant firms in the field. For this purpose we are also using two web tools of Google which help in the Search Engine Optimization (SEO), Google Webmasters Tools and Google Analytics. These two tools are not magic wards that will drag your website to the first page of Google results, but with the right use they improve a websites potential to this way. It is obvious at this point that we are going to introduce Google friendly techniques.

In this paper we start with some common used techniques for SEO and some that might improve the accessibility of a website. We talk about the recent change in the Google's search algorithm that changes the way SEO analysts were thinking and sets quality as the indexing "King". Furthermore, we simply explain the usage and the benefits of Google Webmasters Tools and Google Analytics. Moreover we show some experimental results derived from real websites that use these tools. And finally we conclude with our personal idea over these tools and some suggestions.

2 SEO for Google

As SEO, are defined all the actions that a webmaster makes in order to place his/her website as high as possible in the result page of a search engine, always when the user rights a query of specific syntax and/or spelling.

There are many SEO techniques on the internet nowadays. Almost every SEO analyst has created a few by his/her own by trying to understand how Google Search works better for the websites that he/she is responsible. Website directories like DMOZ, social networks like Facebook or Twitter, web advertisements like Google Ads or even spamming by using e-mails, forum messages and comments in popular websites, can improve significantly the traffic of a website.

However it is not our purpose to criticize these techniques in this section, but to present how Google understands SEO. Moreover we present some layout techniques, derived from internet surveys that might help improving the accessibility and the quality of a website.

A metric that made Google the number one search engine is PageRank [2]. Today it remains a mystery whether or not Google is taking under serious consideration PageRank, as there are many discussions on the internet which say that Google have not given a good updated PageRank the last period. Perhaps all these rumors had to do with what Google said "*Today we use more than 200 signals, including PageRank*". Recently, in March 2011, they also updated their search algorithm with the Panda algorithm, which impacts 11.8% of the search queries [5] and intends to quality search results. It is currently noticeable in US only, but in time will be applied globally.

As we mentioned above the new algorithm indexes better quality websites. This means that websites that don´t try to trick their way to the top of search engine results and provide the users with a better web surfing experience will be ranked higher [6]. Of course we are not discussing ways to write unique content or how to be creative in design, but how they can have a better ranking by understanding the contents structure initially and what they can do to improve it.

In research made by Brandon Falls et. al [1] they tried to review 100 results of the Google products engine. What is interesting for us is that in the on page optimization part of the research they found out that half of the websites need improvements in header tags or Logos [table1]. For optimizing quality but also identifying ways to improve the user experience on a website, Google provides two free to use tools, Webmaster's tool and Google Analytics.

Topic	Products Passing	Grade
Heading tag use	**68%** (61/90)	Satisfactory
<h1> tag use	**43%** (26/61)	Needs Improvement
Logo image link destination	**39%** (38/97)	Needs Improvement
Logo image alt text	**58%** (57/99)	Needs Improvement
Descriptive internal anchor text	**67%** (67/100)	Satisfactory

Table 1 – On-page optimization [1]

3 Google Tools

As mentioned in the previous section Google created two tools that can improve the quality of a website, but how can an inexperienced webmaster start using them and extract some valuable information, will be presented in this section as well as in section 4 where experiments on real websites are presented.

The very first thing that a new webmaster, with his/her website only a few minutes online, is to check how to it appears in Google Search

and then is when the first disappointment comes. The second one comes when webmasters realize that the visitor's counter, which they have implemented, does not provide much information about the visitors, new or old ones. For the first task webmasters can use the webmaster's tool, while for the second one the Google analytics. However, this might only be the very first reason to use these tools as their functionality extends way more than this.

3.1 Webmaster's Tools

What is it – Webmaster's tools is a collection of web based applications, provided for free by Google, that help webmaster's to improve the quality of a website as it matters the content and the technical part.

How does it work – The webmaster has initially to add the URL of the website. For the added URL he has to select a type of verification. He can place a script that the tool is generating, inside the body tag of the website's html code or if he already has a Google analytics account, to verify his URL through this.

What does it offer – Webmaster's tools offers a variety of applications and reports, which we are going to present in this section.

After adding your website and verifying it, the first page that you will notice is the Dashboard. The Dashboard is a summary of a few sub-tools that provide you with some information such as sitemaps and crawl errors.

Sitemaps: The first thing a webmaster should do is to add his sitemaps. The sitemap protocol [7] helps the web crawlers to better retrieve information from a fast growing web. Usually sitemaps are formed in XML format and they are translating the website's structure to Google. Webmasters tool has a 500 URLs limit in the URLs submitted. However, allows webmasters to upload more than one sitemaps.

Crawler access: Since the crawlers are more than one, there is need for rules that will instruct the crawlers how to crawl a website [8]. The robots.txt is an instructions document that regulates the crawlers accessing a website.

Websites might have pages that the webmasters do not want Google to index (Login private areas, RSS feeds or crucial data). Webmasters can block these sensitive areas by writing a robots.txt and placing them in the root directory of their websites. Google in order to help with this procedure provides 3 useful tools "Generate robots.txt" that creates the txt file, "Remove URL" that can block a page or directory and the "Test robot.txt" that can simulate the access of the Google crawler on a website. More particularly, the robot.txt is a necessary component of a quality website, due to the fact that better crawler accessibility implies better search engine indexing. Note that with robots.txt you do not block only pages for the Google search, but for any web search engine.

Sitelinks: Webmasters have not many choices dealing with the sitelinks. The sitelinks are inner links that are generated automatically by Google. The webmaster has the choice to block the ones that they seem him/her not appropriate.

Additionally to these tools, some interesting reports are available, as we mentioned above.

Site Performance: In the second section we talked about the new algorithm of the Google search. We said that the user's experience is also one of the quality attributes for better indexing. The speed of loading a website is considered as a fact that can impact the user's experience. In the Site Performance menu, it is presented the site's loading time. Faster websites are offering better user's experience even for the ones that are using slow internet connections.

Search Queries: The Search Queries report is the most important as it matters the content and the keywords of a website. It shows the search queries that have returned pages from the website, with which keywords, and the improvement and significance of these keywords through time. This report can help people understand which keywords fit better to the websites description and thus can improve the websites traffic through search engines. A detailed example about the usage of this report in a real website is showed in the "*Expirement #1*".

3.2 Analytics

What is it – Analytics is a web based tool offered by Google in which you may view different statistics about a web site. It shows how visitors use a website, and by this way someone can change the website's design depending on what users want. In combination with the Webmaster's tool, it can also increase the visits of the website, by providing useful information.

How does it work – Firstly it has to be activated. It activates by placing a script under the <head> tag of a website's html code, as exactly with the webmaster's tools. In case that the website is using a separate html file for every page, the script must be placed in all the files. Google´s script will be triggered every time a user enters the site and gathers valuable information. As the company assures, the data collected will not be shared with any third party company. Have also in mind that Google does not collect personal information about the users that visiting a website.

What does it offer – Google Analytics offers a huge variety of reports and statistics and it even lets webmasters to make their own customized reports.

Visitor Reports:

1. Visits: The number of visits the website has had, per time period. There are also unique visitors which are visitors that came once to the site in time period. The problem with this is how does Google know the visitor is unique? They do it by placing a cookie on the visitor's computer. But if the cookie is deleted the visitor will be recounted.
2. Page view: The quantity of pages in the website that where viewed in a time period.
3. Time on site: The amount of time users stay active in the website in average
4. Bounce Rate: The number of users that leave the website after visiting the first page.
5. Map Overlay: Shows from which countries the visitors are native from.
6. Languages: Which languages the users speak.
7. Browser Capabilities: Of each user it identifies the Operating System, the browser, JavaScript enabled, flash enabled etc.
8. Mobile: From which mobile device someone accessed the website.

Traffic sources: These reports show from where the user came. The traffic source might be a search engine like Yahoo! and Google, an advertisement, some other websites that back-links to the specific one, or direct (used the browser bar). Also it tells us about the browsers used to access our site and what screen resolution is most popular. Analytics retrieves the data from the computer which the visitor is using, so if the language of the computer is English but the visitor speaks French as native language we will not know this. The same counts for the country of origin, since it's not really the country of origin of the person but the country from which he/she entered the site.

Content: In these reports it shows which pages in the website the user explored, where he went out and time on each page. Also which page the user entered first and where he traveled next.

One interesting service provided here is "In-page Analytics" which provides the webmaster a visualization of the site and which links are mostly clicked.

Goals: Goals are a way of measuring how many times a visitor does something you wanted him to do. For example, if you wanted users to access a certain page after viewing certain pictures in your

gallery, or, how many people accessed the reservation page in hotels web site.

Alerts: An alert is simple a notification about something that has happened on the website, you may choose to receive these notifications through an email. For example, you want to be notified when a user lasts more than 5 minutes on the website.

All these reports can be combined in the custom reporting option, so visualization of the data completely users choice, they may also be compared in the time line. These reports may be applied to more than one websites; Analytics lets you add as many websites as you want to this tool.

For most graphs there is also the option of motion charts, which flow in a period of time and explain the change of pages through time.

Interpreting data – The most important aspect of using this report is interpreting their data correctly, and maybe this task is independent to the site since each site has its own goals and targets. For example a blog site about a country has a geographical target for that country and does not really care about visits from other countries.

Tables – Let us start with some basics, the visit count against the bounce rate, which is influence by the content of the site and how it is displayed. As we explained before a bounce from your site is a visit in which the visitor did not proceed to any other pages within the site and left it before 30 seconds of being there.

Day	Visitors	Bounces
Mar 14, 2011	32	13
Mar 15, 2011	23	5

Table 2 –Google Analytics bounce vs. visitor table

Most people think that by having a bigger visitor's number the website has more popularity and it is better build than if it is having fewer visits. This might be true in some cases, but in Table 2 we see that the visits in March 14th is greater than in March 15th and still the amount of people that bounced from the site in the 14th is 13, so actually the site only got 19 real visits against 18 from the 15. Does this mean the 14th had 1 more useful visit, or that the 15th had 8 less unproductive visits? In our point of view it's more valuable the ones you lost than the ones you gained.

To read correctly the statistics presented it is very important to take in account all measurements before drawing conclusions.

Time lines - The data in Google Analytics is mostly graphed in time lines; these graphs have the possibility to compare different time periods, so for example you would like to compare January 2011 with December 2010, view fig. 1.

fig. 1 – Google Analytics Time line

The time lines looks very nice, but they do not tell much, this is why under the graph we see a much clearer comparison per stat. To change the time line to any of these stats we simply click on them, for example observe that as from January the Bounce rate has gone down 35% which is very good.

Custom reporting – Google Analytics offers as a customizable tool, after all, businesses have different goals in what's important for them.

4 EXPERIMENTS

In the section 3 we introduced the tools. In this section we are going to use the tools on real websites and derive some valuable information for their improvement, through a number of experiments.

Experiment #1, **rating keywords:** In this experiment we are using the Webmaster's tool and specifically the Search Queries report. The experiment was on a Greek blog, for this reason most of the queries in fig. 2 will not be readable from the majority of the readers. However, the letters are hard to read due to the size of the picture. The most important is the numbers, which fluctuation we are going to explain.

We selected 15 of the queries. A few of those there were already in the description of the website, but most of them were not.

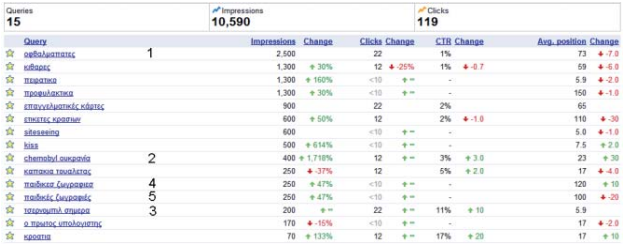


fig. 2 –Search Queries

The query "n.1" is a keyword in the websites description. What we mention here is that although it has 22 clicks, these clicks are coming from 2.500 impressions. This means that 2500 people saw this result but only 22 clicked it. Immediately, it makes us think that this keyword should be better removed or replaced by another which is more representative of the website's content.

A second observation has to do with the query "n.2". This query had to do with a specific article. This article fast attracted visitors and as the fig.2 shows 3% of the impressions was turned into visits. For going further with this observation, two more articles were written about this subject and the first part of the query changed in Greek, query "n.3" and placed in the description of the website. In just a week 11% of the impressions turned to visits and actually quality visits, which brought the query in 6th position of the search results (5.9 Avg. Position). This means that in every query the website is shown on the first page of Google search results, which might be translated with potential higher traffic.

Finally, our last observation had to do with another keyword of the blog's description. Queries "n.4" and "n.5" represent exactly the same phrase with some minor notation differences. Although, combined they have 500 impressions, they have less than 10 clicks, which means not enough information to classify them. This observation bring us to the result of the first observation, when was mentioned that the keyword should be possibly removed or replaced.

Experiment #2, **bounce rate improvement:** Table 3 shows the result of some minor changes we applied, which we will explain in more detail.

Day	Visitors	Bounce Rate	Bounces
Mar 17	29	13%	4
Mar 16	28	25%	8
Mar 15	23	18%	5
Mar 14	32	38%	13
Mar 13	28	39%	13

Mar 12	26	34%	11
Mar 17	29	13%	4
Mar 16	28	25%	8
Mar 15	23	18%	5

Table 3

The Visit number of each date is approximately the same, and it is a good number for the web site in question. But we did see that the bounce rate of our site was too high, we had to do something. The reason this happened so often was unclear, so we decided to view other web sites related to ours that had a good page rank, and we found out that by placing some text of what the site was about or what you could do in it and links to these services or activities in the index page would make the users get much more interested in the site.

This also meant the site had more "useful" impressions by Google searches, by useful we mean that It appear in the search for people who were looking for something related to the site.

In conclusion, your start page is the most important page in your site; it will determine how many surfers will bounce off and how many will not. Make it interesting; place some fast and understandable text, use images with proper tagging, place links within your site (not flash based, we will explain this latter on) and use as little adds or commercials as you can.

Experiment #3, **site esthetics and colors:** The time line in fig. 1 shows how the site has changed its statistics from December to January.

The colors and design of a site will reflect what the company is all about, users respond directly to elegance, neatness and aggressiveness. Our site is a hotel site in the mountains, the moment you read this you probably thought of green. The original designer decided to make it with orange vivid colors, and we decided to change this to more dark green colors in the back to cool green colors in the front. The new site was launched as from the first of January 2011, and immediately it made spectacular changes in stats.

Neatness and elegance, this is completely dependent on who is viewing the site, but for most users a well constructed site makes them much more comfortable. What does "well constructed" mean? Elements should not surpass others in width if they belong to the same layer, menus should be readable don't place all your links cramped up, make subcategories, and as a maximum set 5-6 head categories. It is always a good idea to have white background to text and images areas, but a light color could also work. Keep content and colors in different pages to the title of the page, use navigational links, for example; if the content talked about a certain tour link this tour to that text.

Aggressiveness, this also depends on your site, but in general, do not use capital letters in body text,on the other hand menus could make use of them. The body font size should not be more than 14px big and use bold only for important words and titles. Do not use highlighted text; it will look desperate for public. Colors must not be blinding or blinking, so don't use sharp reds, greens or yellows.

Experiment #4, removing flash: It is not that we hate flash; it just that Google does not like it. The truth is that Google announced in 2008 that it could identify text content follow links in flash websites, but it is not fully SEO friendly [9]. The reason is that flash consists of animated elements in one page which the Google-bot will not read properly, so images in these elements for example, will be ignored by Google.

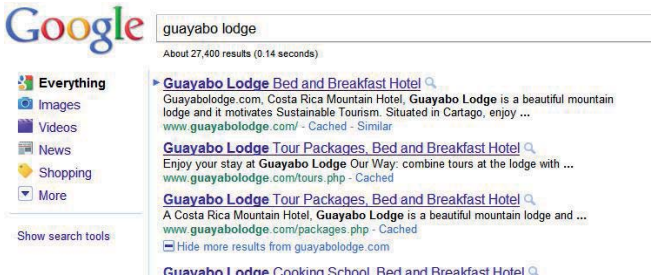


Fig 3 – Google search results

Fig. 3 shows that the first 8 results on Google with the above key words are of our site; this is not very impressive since guayabolodge.com is our site domain. But normally the results of typing the sites name are one or two, then why did we get 8? It is simple, indexing within the site is very important for Google. In our old site the designer had placed the links in a flash, making it impossible for the Google-bot to read these links. So we changed our links to plain html and indexed them in the sitemap.xml and ror.xml pages for Google to index them correctly. The result after some weeks for Google to index the correctly was incredible. This also goes for images, do not place them into a flash animation, and try using other technologies like JQuery.

Experiment #5, adapt to the circumstances: More than an experiment it is to make the web page meet the needs to most of the users. We changed our site depending on what most of the user's use, like the browser, the screen resolution, if they had flash and JavaScript enabled.

Browser	Visits	Visits %
Internet Explorer	437	44%
Firefox	365	36%
Chrome	104	10%
Safari	85	8%
BlackBerry9100	1	<1%
Mozilla Compatible Agent	1	<1%

Table 4 – Google Analytics browser reports

In table 4 we observe that IExplorer is the most used browser within users that access the website, for this we adapted the site for IExplorer and then took in account FireFox´s standards, we also tested the site in Google Chorme.

Screen Resolution	Visits	Visits
1024x768	162	16%
1280x800	162	16%
1280x1024	150	15%
1440x900	87	8%
1366x768	85	8%
1600x900	55	5%
1920x1080	49	4%
1024x600	30	3%
1680x1050	24	2%
1152x864	21	2%

Table 5 – Google Analytics Screen resolution report

In table 5 it is clear that most users have a 1024x768 screen resolution, so when designing the new site our maximum content width was 900px.

5 Conclusion

As it is mentioned in the introduction, it is particularly easy to create and maintain a website in our days. The difficult part is to target the appropriate audience. In our paper we discussed the way to do this through the search results of a search engine and specifically, the most used search engine, Google search.

Google offers tools to improve websites and make you aware of what is happening in them. Websites are virtual hang outs and information centers, make visitors interested and they will invite more friends to check out the site. However, this will not affect its position in the Google search results. What will change that is the way that a website fulfills the Google search requirements. We talked about the changes in the Google search algorithm and the quality characteristics [5] that affect the result. We introduced the two Google tools that are capable of improving the quality of a website. And we finally showed, through a number of successful experiments, how to improve the content, the target and the layout of a website and thus its quality.

Although, uniqueness, creativity and attractiveness are the most important ingredients for a popular website, using these tools in the way that we introduced, a less popular website could see a significant improvement in visits, which can be respectively satisfying whether it is a personal or a business website.

In our experience site development is very complicated, but if you want to make it worth we recommend using these tools as a reference on how to make the site much better.

References

[1] Brandon Falls, Andi Goradia, Charlene Perez. Google's SEO report Card. March 1 2010..
[2] A.N.Langville and C. D. Meyer, Google's Page Rank and Beyond.
[3] Avanish Kaushik. Google Analytics uses in ecommerce and other web applications, second edition.
[4] Justin Cutroni, Google Analytics description manual book.
[5] Finding more high-quality sites in search. http://googleblog.blogspot.com/2011/02/finding-more-high-quality-sites-in.html (retrieved in April 2011)
[6] Webmasters guidelines. http://google.com/support/webmasters/bin/answer.py?answer=35769 (retrieved in April 2011)
[7] Uri Schonfeld, Narayanan Shivakumar. Sitemaps: Above and Beyond the Crawl of Duty, WWW 2009 Madrid
[8] Santanu Kolay, Paolo D'Alberto, Ali Dasdan, Arnab Bhattacharjee. A Larger Scale Study of Robots.txt, WWW 2008 Beijing
[9] Eric Enge, Stephan Spencer, Rand Fishkin, Jessie C. Stricchiola. The Art of SEO, Mastering Search Engine Optimization, O'Reilly 2010

Digital Image Forensics

Jan Kazemier and Michiel Heijkoop

Abstract—Digital photography has made image manipulation (i.e. "photoshopping") increasingly easy and common. However, the authenticity must be relied on in legal proceedings and journalism. Detecting whether or not a digital image has been manipulated is a difficult task. In this paper we will look at several of these techniques to demonstrate the state of the art. We will show that while certain methods can provide strong indicators of manipulation, a definitive answer cannot be given.

Index Terms—Digital image, analysis, forensics, photoshop.

1 Introduction

As Popescu and Farid [18] explain, a digitally altered photograph, often leaves no visual clues of having been tampered with. This means it can be indistinguishable from an authentic photograph. As a result, photographs no longer hold the unique stature as a definitive recording of events.

Ashwin Swaminathan et al. [21] say that with increasing popularity of digital imaging and the availability of low-cost image editing software, the integrity of digital image content can not longer be taken for granted.

This poses an interesting challenge for the field of image analysis to find and propose techniques to detect such image tampering. This can, for example, be important in legal proceedings where a digital photograph can used as evidence.

Therefore, there is a need for a method to reliably determined whether or not an image has been tampered with.

Using watermarks[13] would be a good way to ensure an image is authentic, but is not feasible in practice as most cameras do not provide this functionality. This is unlikely to change in the short term.

Several researches propose different approaches for detection. In this paper we will illustrate and explain several different techniques described in literature. Some other techniques are listed under references.

2 Intrinsic Fingerprints

Ashwin Swaminathan et al. [21] describe a technique detecting traces left by image processing processes, called "intrinsic fingerprints". Their proposed method is based on the observation that many processing operations, both inside and outside acquisition devices, leave distinct intrinsic traces on digital images, and these intrinsic fingerprints can be identified and employed to verify the integrity of digital data.
In their paper they propose a novel mothodology for digital image forensics of color images. They present techniques to identify which traces are left behind in the image when it goes through different stages in the information processing chain. These specific traces are here reffered to as the "intrinsic fingerprints".

2.1 Intrinsic fingerprint estimation of in-camera processing

First an estimation of the intrinsic fingerprint as a result of the in-camera processing by a detailed imagaging model and its component analysis is performed. The intrinsic fingerprints caused by postcamera operations is estimated by a model of a manipulation filter. Swaminathan suggests that the absence of camera-imposed fingerprints from an image indicates that the image is not a camera output and is possibly generated by other image production processes. Any change or inconsistencies among the estimated camera-imposed fingerprints, or the presence of new types of fingerprints suggest that the image has undergone some kind of processing after the initial capture [21].

- *Jan Kazemier is a Computing Science student at the University of Groningen, E-mail: j.kazemier@gmail.com.*
- *Michiel Heijkoop is a Computing Science student at the University of Groningen, E-mail: mheijkoop@gmail.com.*

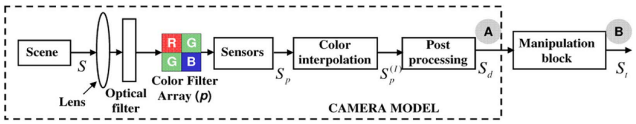


Figure 1: Image acquisition model in digital cameras [21].

2.1.1 Image Acquisition

Most digital cameras use a color filter array (CFA) to sample the real-world scene [21]. In figure 1 the system model of the digital image acquisition is shown schematically. The CFA consists of three color sensors (red, green and blue), which capture the real color of the real-world schene at the corresponding pixel location. This yields a 3D array of the size $H \times W \times C$, where H and W denote the height and with of the image in pixels, and C denotes the number of captured colors, three in this case. After capturing the color values, intermediate pixel values are interpolated using the neighboring pixel values. Then the three images corresponding to the three color components go through a postprocessing stage. Here, depending on the camera, the images may undergo different processing operations [2], [1], which might include white balancing, color correction, gamma correction, lens vignetting correction, lens distortion removal, denoising, etc.

2.2 Estimating Camera Component Parameters

As every step in the acquisition uses different algorithms, that may be particular to the camera manufacturer, brand, or mode, they leave intrinsic fingerpint traces on the output data. In an earlier work, Swaminathan et. al [20] describe methods to estimate these in-camera fingerprints from outputs from the camera model. The CFA pattern and the color interpolation coefficients can be jointly estimated from the output image [20] in the following way. For every CFA pattern p in the search space P, linear models are fitted, in order to compute the interpolation coefficients in different types of texture regions. The image is divided into three types of regions based on the gradient features in a local neighborhood. We find the horizontal gradient ($\mathscr{H}$, Equation 1) and vertical gradient ($\mathscr{V}$, Equation 2) using simple gradient filters on the image, yielding the following functions:

$$\mathscr{H}_{x,y} = |I_{x,y-2} - 2I_{x,y} + I_{x,y+2}| \quad (1)$$

$$\mathscr{V}_{x,y} = |I_{x-2,y} - 2I_{x,y} + I_{x+2,y}| \quad (2)$$

Where $I_{x,y}$ is the pixel value in the output image at position (x,y). The image pixel at position (x,y) is classified into one of the three regions:

- Region $\mathscr{R}_1$ contains those parts of the image with a significant horizontal gradient (i.e. $\mathscr{H}_{x,y} - \mathscr{V}_{x,y} > \mathscr{T}$), where $\mathscr{T}$ is a suitably chosen threshold;

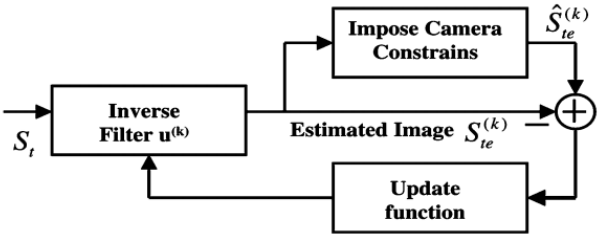


Figure 2: Recursive algorithm to determine estimates of the coefficients of the manipulation filter [21].

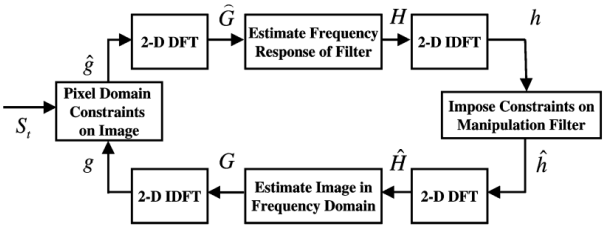


Figure 3: Estimating the coefficients of the inverse manipulation filter iteratively [21].

- Region $\mathscr{R}_1$ contains those parts of the image with a significant horizontal gradient (i.e. $\mathscr{V}_{x,y} - \mathscr{H}_{x,y} > \mathscr{T}$), where $\mathscr{T}$ is a suitably chosen threshold;
- Region $\mathscr{R}_3$ contains the remaining parts of the image; primarily smooth regions.

Using the final camera output, a set of linear equations for all the pixels in each region is obtained and solved to obtain the interpolation coefficients, which are called $\alpha\mathscr{R}_i$. Once estimated, they are used to reinterpolate the image and and the interpolation error. The CFA pattern that gives the lowest error gives the estimate of the CFA pattern. The estimates are also shown to be robust to moderate levels of postprocessing operations, such as JPEG compression, and white balancing done inside the cameras [20].

2.3 Estimating intrinsic fingerprints of postcamera manipulations

The analysis of intrinsic fingerprints caused by postcamera manipulations is built upon the component forensic analysis of the previous subsection. Swaminathan et al. [21] propose methods to identify whether an image has undergone any further processing after it has been captured using a digital camera. Here, they mainly focus on images that constitute in a bulk of camera-captured images.

First it is assumed that the image is manipulated after in-camera post-processing, corresponding to the point B in Figure 1. Then the postcamera processing steps are represented as a combination of linear and nonlinear operations, and are approximated with a linear shift-invariant filter. Using blind deconvolutions, the coefficients of these manipulation filters are estimated. Those coefficients serve as postcamera fingerprints to answer forensic questions, e.g. is the image authentic, and what is the origin?

An estimate of the camera output is obtained through passing the given test image through an inverse manipulation filter u. The coefficients of this filter are estimated by solving an optimization problem that minimizes the camera model fitting error.

The filter coefficients of the manipulation filter can be directly estimated in the pixel domain through a recursive procedure, as shown in Figure 2. The iterations are started by setting $u^{(0)}$ to be a delta function; corresponding to direct camera outputs. In the kth iteration, an estimate of the camera output $S_{te}^{(k)}$ is obtained by passing the test image through the estimate of the inverse blurring filter u. The recursive procedure is repeated for a finite number of iterations or until convergence occurs.

Since the performance estimating the coefficients of the inverse manipulation filter depends on the size of the averaging filter, and would ideally would require an infinite length kernel for it's inverse, Swaminathan et al. [21] propose a solution to find the estimate directly in the frequency domain, using an iterative blind deconvolution method, as described by Ayers et al. [3]. A schematic representation of this method is shown in Figure 3.

The frequency response of the manipulation filter for an unmanipulated camera output suggests minor deviations from an ideal flat spectrum. The frequency domain coefficients are determined, and similarity between the coefficients of the test input and a reference image is computed. The test input is then classified as unmanipulated if the similarity to the reference pattern is greater than a suitably chosen threshold. If the input image on the other hand has been manipulated, the estimated manipulation filter coefficients would be different: it would include the effects of both the postcamera manipulation operations along with postinterpolation processing inside the camera and therefore the similarity score would be lower than the chosen threshold.

Canon Powershot A75
Canon Powershot S410
Canon Powershot G6
Canon Powershot S400
Canon Powershot S1 IS
Canon EOS Digital Rebel
Nikon E4300
Fujifilm Finepix S3000
Sony Cybershot DSC P72

Table 1: Camera models used in experiments in Swaminathan et al. [21]

2.4 Detecting manipulations on camera-captured images

Swaminathan et al. [21] have tested their methods on 900 images of 512×512 pixels, which are randomly cropped portions of images taken with nine different camera models (see Table 1). These images were then processed to generate 21 tampered versions (see Table 2 per image to obtain 18900 manipulated images. Figure 4 shows the frequency response of the manipulating filters for camera outputs.

The tests are performed on two hypotheses:

- Γ_0: image is a direct camera output
- Γ_1: image is not a direct camera output and is possibly manipulated in some way

The performance of the threshold based classifier is examined in terms of the receiver operating characteristics (ROC). An ROC graph is a technique for visualizing, organizing and selecting classifiers based on their performance [8]. For more information on how to read ROC-graphs, we refer to Fawcett [8], as this is outside the scope of this paper.

For each original image the frequency response of the manipulation filter is computed. Then the similarity with the reference filter pattern is measured. The false alarm probability P_F is given by the fraction of original images which similarity score is lower than a threshold τ. Similarly, the probability of correct decision (P_D) is given by the fraction of manipulated images with a similarity score that is less than τ. This process is repeated for differen thresholds τ, until we arrive at the ROC Figure 5 shows.

For each image, the frequency-domain coefficients are computed of the estimated manipulation filter and determine its similarity with the chosen reference pattern. Images with a similarity score that are greater than a chosen threshold are classified as authentic. To choose the reference pattern, a set of N_t training images is selected, along with

Operation	Parameters	Number of filters
Spatial averaging	Filter orders 3-11 in steps of 2	5
Median filtering	Filter orders 3, 5 and 7	3
Rotation	Degrees 5, 10, 15 and 20	4
Resampling	Scale factors 0.5, 0.7, 0.85, 1.15, 1.3 and 1.5	6
Additive noise	PSNR 5dB and 10 dB	2
Histogram equalization		1

Table 2: Manipulating operations included in experiments in Swaminathan et al. [21]

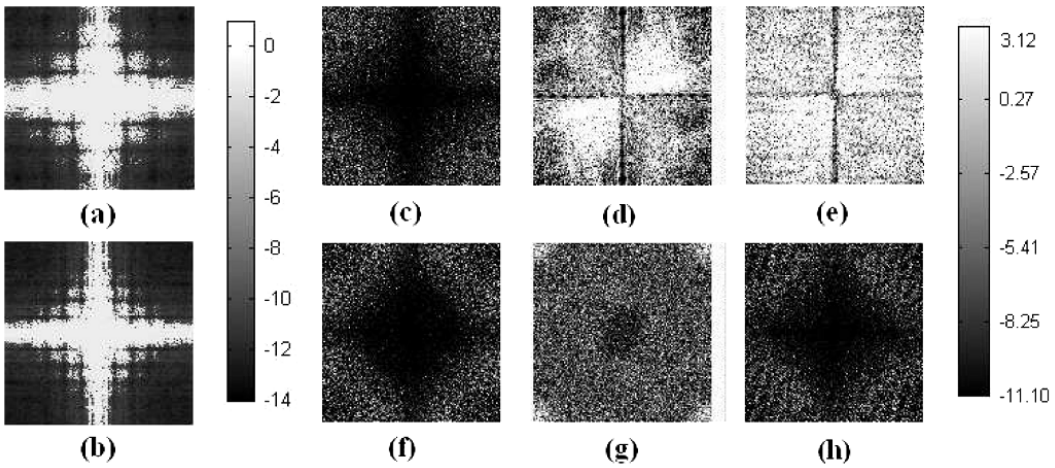


Figure 4: The frequency response of the manipulating filters for camera outputs. (a) 7×7 averaging filter, (b) 11×11 averaging filter, (c) 7×7 median filter, (d) 20° rotation, (e) 70% resampling, (f) 130% resampling, (g) noise addition with PSNR 20 dB, and (h) histogram equalization. (In log scale and shifted) [21].

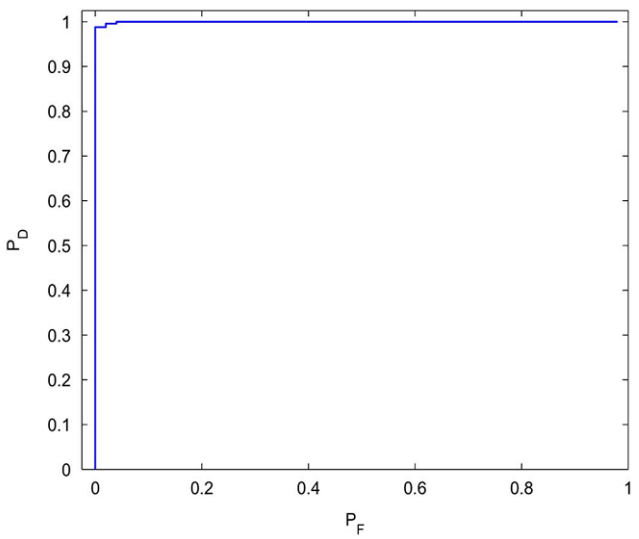


Figure 5: Preferred ROC for distinguishing between simulated camera outputs and their filtered versions [21].

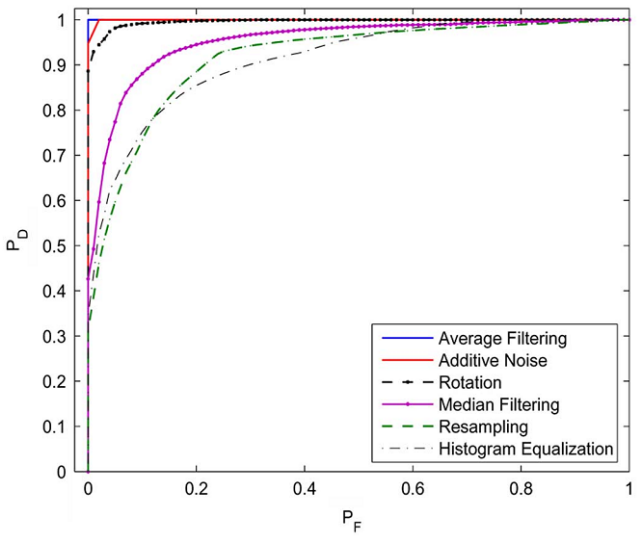


Figure 6: ROC for detection of manipulated images from a Canon Powershot A75 [21].

its manipulated versions in the training stage. Using a threshold τ, the fraction of direct camera outputs with a similarity score lower than the threshold is computed to give the false alarm probability $P_F = Pr(\Gamma_1|\Gamma_0)$, and the fraction of manipulated images with a similarity score of less than the threshold τ is found to give the probability of correct decision $P_D = Pr(\Gamma_1|\Gamma_1)$. This process is repeated for different thresholds to arrive at the ROC, and compute the area under the curve.

The performance of the threshold-based detector was measured and averaged over 100 iterations. the corresponding ROC for detection of manipulated images from a Canon Powershot A75, where 50×21 images were used for training and the remaining 50×21 images were used for testing is shown in Figure 6. At a relatively low P_F around 10 percent, the probability of correct detection is about $80-95\%$ for most types of manipulations tested [21].

Likewise, the ROC for detection of manipulated images from all images in the database, where 200×21 randomly chosen images were

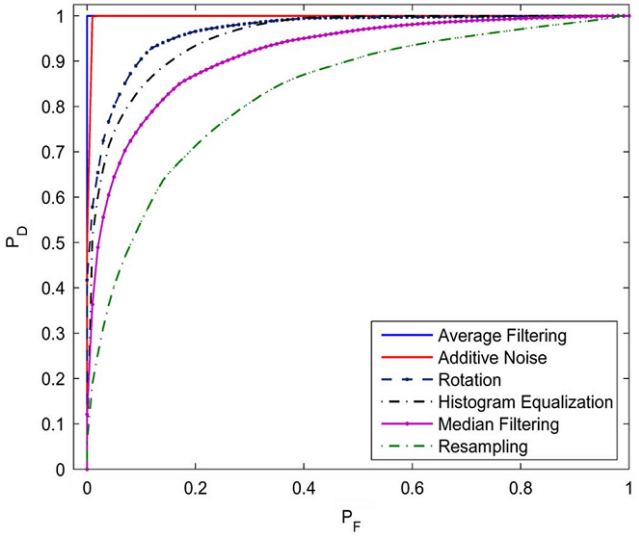


Figure 7: ROC for detection of manipulated images using all images in the database [21].

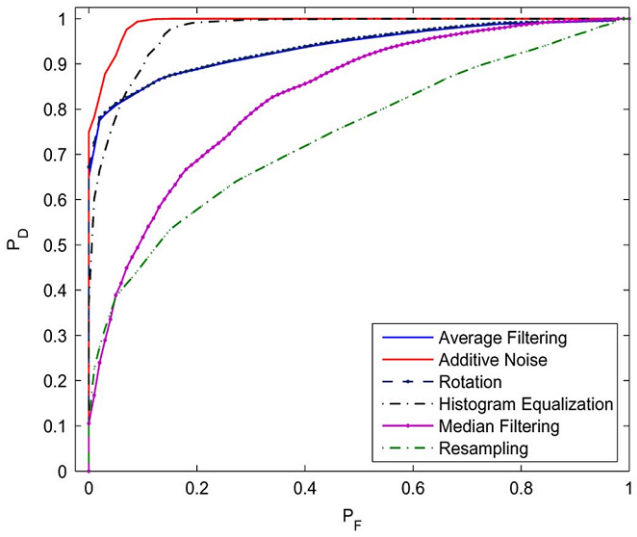


Figure 8: ROC for detection of manipulated images using different cameras for training and testing [21].

used for training and all remaining images were used for testing is shown in Figure 7. These images were captured under the default camera settings and may have undergone different kinds of in-camera postprocessing operations, such as JPEG compression after color interpolation [21]. In this case, for P_F close to 10 percent, a probability close to 100 percent is observed for such manipulations such as spatial averaging and additive noise. Around 70 to 80 percent correct detection is observed for median filtering, histogram equalization and rotation. These results are better than the results described in [17] and [7].

Since the proposed techniques do not require the images to be from the same source camera, Swaminathan et al. [21] demonstrate that different sources can be used. The performance results, averaged over 100 iterations in Figure 8 show the performance of the proposed technique using 100 images from the Canon Powershot A75 for training and 100 images from Sony Cybershot DSC P72 for testing. The figure shows that the performance is good for most manipulations - a detection rate of 80 to 90 percent is obtained for P_F around 10%. This result is comparable to the plots in Figures 6 and 7. The drop in performance for some manipulations, such as resampling, can be attributed to the absence of the original camera make/model in training [21].

3 Recognizing JPEG-compressors

In order to identify whether or not an image has been manipulated, it is useful to know what software or hardware has been used to perform the JPEG-compression. If it can be shown that an image has been compressed to JPEG using a tool like *Adobe Photoshop*, it cannot be guaranteed that the image is authentic. One way of doing this, is looking at the quantization table that has been used to compress the image.

3.1 JPEG compression

To explain this technique we must first give a short overview on how JPEG compression is defined, based on Wallace[22]. It is illustrated in Figure 9.

JPEG encoding consists of several steps. For color images, an uncompressed bitmap image is first converted from RGB color space to YCbCr. This means each pixel will have a value for its luminance (brightness) and two chroma values that define the color.

This conversion allows reduction of the image resolution in the color domain while maintaining the original resolution in the brightness component. This is because the eye is more sensitive to

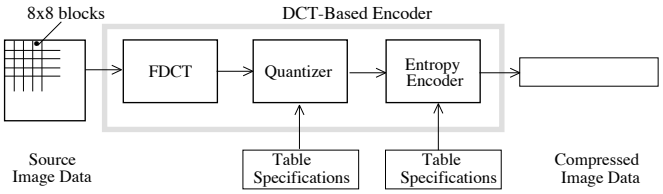


Figure 9: The steps in JPEG-encoding as illustrated by Wallace[22].

differences in brightness than it is to color. However, this step is optional and we will ignore it in our analysis.

Then, pixels are grouped in blocks of 8×8 pixels and the image is transformed to the frequency domain with a discrete cosine transform (DCT).

In the frequency domain, the image is divided by a matrix – called the *quantization table* — that defines a coefficient per frequency. The resulting values are rounded to the nearest integer. In this step information is can be lost, as frequencies with a small coefficient will be rounded to zero.

Finally, the resulting data is entropy-coded for additional compression, no information is lost in this step.

3.2 Quantization table analysis

As explained in the previous section, the amount of information lost in JPEG encoding is largely dependant on the quantization table used in the compression process.

Knowing the quantization table used to compress an image can give a very strong clue on what software or camera was used. Farid[6] experimentally determined that 62 out of 204 cameras had an unique quantization table. On average, each camera shared its quantization table with 1.43 other cameras, often a camera with the same manufacturer.

Adobe Photoshop allows the user to select a quality level when exporting an image as a JPEG. The choice of one of the twelve built-in quantization tables in Photoshop is based on this selection[14]. If one of these tables is found in an image, it is most likely saved using Photoshop and therefore possibly manipulated. For example, this could mean that the image should not be used as evidence in court.

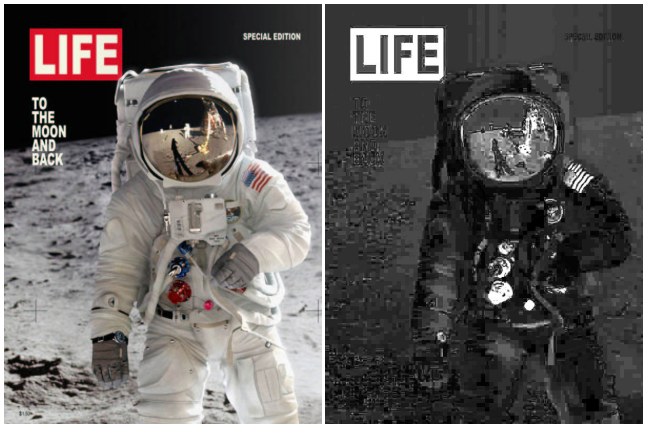


Figure 10: A magazine cover that is composited from different sources.[15]

3.3 Adaptive tables

Kornblum[14] found that often cameras do not use a single quantization table, but can have a mechanism for adaptively scaling the table based on the image. This means that in order to correctly identify the origin of a picture, a large number of potential tables per camera model would have to be known. An identical table might also be produced by both a camera and certain manipulation software, making analysis inconclusive. Rather than identifying the exact origin of an image, Kornblum suggests to only eliminate images that are suspect (i.e. images that have a quantization table that is possibly generated by manipulation software).

4 JPEG Error Level Analysis

In Luo[16] and in Krawetz[15] techniques are proposed that compare the number and type of errors generated by various steps in the JPEG compression process to show if an area (as small as one block, or 8×8 pixels as explained in Section 3.1) has been compressed multiple times. This can be a strong indicator of manipulation.

If parts of an image have been compressed a different number of times, or with different quality settings, it could indicate that these parts were added to the image from another source.

4.1 Using Principal Components Analysis

If a section of an image has been compressed differently than the rest of the image, it will exhibit a different rate of JPEG artefacts (or: errors). However, these artefacts may not be clearly visible. Krawetz[15] propose to use Principle Components Analysis, to make these artefacts more clearly visible.

Principle Components Analysis (PCA) is a means of clustering data dimensions. A grey-scale image can be represented as a 3D field in which the intensity of the pixel value is the third dimension. The first principle component will be the plane of data points with the most variance[12].

Krawetz demonstrates that by visualizing the first principle component, artefacts are significantly easier to observe than in the original image. This allows us to more clearly notice differences in the amount of artefacts in various regions of the image and provide clues as to what parts of the image were not originally present.

In Figure 10 a real-life magazine cover is shown that has been spliced from multiple sources. Krawetz notes that it can be seen by the artefact rates that the background is from a different source image than the space suit. The reflection in the helmet is also rendered from a different source.

4.2 Recompressing

Using principal components analysis is based on the fact that a after a certain number of times, recompressing an image will no longer change the outcome by further degrading the image.

(a) A child in an inappropriate shirt (b) Difference image

Figure 11: Krawetz[15] demonstrates that the print on the childs t-shirt has been manipulated

Parts of the image that have been manipulated might have originally been saved using different compression settings or even a different amount of times. These parts can take a different number of recompression steps in order to reach this stable state than the rest of the image.

By recompressing the image multiple times and plotting the difference between the image and its recompressed copy, degradation caused by the compression algorithm can be observed. These difference plots can provide strong clues as to what areas might have been manipulated. An example is found in Figure 11.

5 Statistical measures

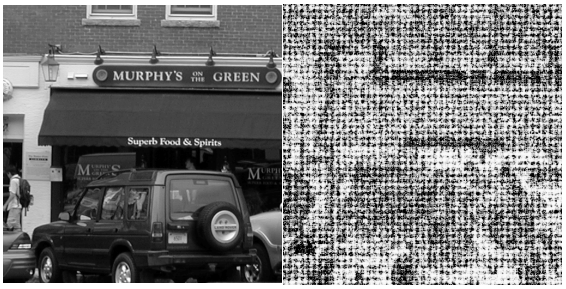
Popescu and Farid[18] propose using statistical measures to detect traces of digital tampering. When an image is manipulated, often operations like rotation, brightness adjustments, blurring or adding noise are applied to parts of an image, to mask manipulation.

5.1 Detecting re-sampling

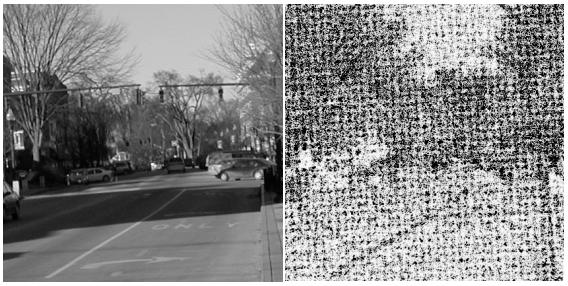
When adding a foreign object to an image, it will often have to be resized and/or rotated to match the scale and orientation of the target image. This is conventionally done using resampling. In resampling, the target resolution or orientation is simply overlayed on the original. The value for each pixel in the target images is determined by sampling the original. One could simply take the value of the nearest pixel in the original image (nearest-neighbour) or use a combined value from multiple pixels in the neighborhoud (e.g. bilineair interpolation). This yields a new image based on the original with the desired resolution and/or orientation.

This operation introduces periodic correlations between neigbouring pixel values in the target image. As to why, we refer to Popescu[18]. A simple example of this effect is an image that is enlarged (up-sampled) to twice its original size using lineair interpolation. In the resulting image, the pixels in odd rows and even columns will be the average of their two horizontal neighbours. The pixels in the even rows and odd columns will be the average of the vertical neighbours[18].

This kind of correlation can be detected using the statistical expectation/maximization algorithm (EM)[5], an impressive example of resampled images and a plotted correlation map can be found in Popescu.



(a) Upsampled image (b) Correlation map



(c) Rotated image (d) Correlation map

Figure 12: Correlation maps of an upsampled and a rotated image, taken from Popescu[18]

In Figure 12 an upsampled image and a rotated image are shown in which the perodical patterns caused by these operations are clearly visible.

6 Summary and Conclusions

A lot of different approaches exist to aid in detecting image forgeries. However, in all cases a human is required to make final judgement, as it is not an exact science.

Detecting a quantization table that is known to be used by manipulating software can be an easy way of dismissing an image as possibly tampered with, but absence of such a match is no proof of the image's authenticity.

Error-analysis can point a forensic investigator in the direction of suspect areas in an image but do not seem to perform very well in practice. Analysis of intrinsic fingerprints left by a camera requires a large amount of sample data, that might not be available.

A universal solution that will detect manipulation with a high degree of certainty currently does not exist in the field of digital image forensics. Currently, all methods available focus mainly on assisting manual techniques and can only provide clues.

Acknowledgements

The authors wish to thank Michael Wilkinson for expert reviewing.

References

[1] J. Adams, K. Parulski, and K. Spaulding. Color processing in digital cameras. *Micro, IEEE*, 18(6):20–30, 1998.

[2] J. Adams Jr. Interactions between color plane interpolation and other image processing functions in electronic photography. In *Proceedings of SPIE*, volume 2416, page 144, 1995.

[3] G. Ayers and J. Dainty. Iterative blind deconvolution method and its applications. *Optics letters*, 13(7):547–549, 1988.

[4] S. Dehnie, T. Sencar, and N. Memon. Digital image forensics for identifying computer generated and digital camera images. In *Image Processing, 2006 IEEE International Conference on*, pages 2313 –2316, 2006.

[5] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[6] H. Farid. Digital image ballistics from JPEG quantization. *Dept. Comput. Sci., Dartmouth College, Tech. Rep. TR2006-583*, 2006.

[7] H. Farid and S. Lyu. Higher-order wavelet statistics and their application to digital forensics. In *Computer Vision and Pattern Recognition Workshop, 2003. CVPRW'03. Conference on*, volume 8, pages 94–94. IEEE, 2003.

[8] T. Fawcett. An introduction to ROC analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[9] J. Fridrich. Digital image forensics. *Signal Processing Magazine, IEEE*, 26(2):26 –37, 2009.

[10] T. Gloe, M. Kirchner, A. Winkler, and B. Rainer. Can we trust digital image forensics? In *Proceedings of the 15th international conference on Multimedia*, MULTIMEDIA '07, pages 78–86, New York, NY, USA, 2007. ACM.

[11] T. Gloe and B. Rainer. The 'dresden image database' for benchmarking digital image forensics. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 1584–1590, New York, NY, USA, 2010. ACM.

[12] R. Gonzalez and R. Woods. Digital image processing, third edition. *Reading, Mass.: Addison-Wesley*, 2008.

[13] F. Hartung and M. Kutter. Multimedia watermarking techniques. *Proceedings of the IEEE*, 87(7):1079–1107, 1999.

[14] J. Kornblum. Using JPEG quantization tables to identify imagery processed by software. *digital investigation*, 5:S21–S25, 2008.

[15] N. Krawetz. A picture's worth...: Digital image analysis and forensics. In *Black Hat Briefings USA*, 2007.

[16] W. Luo, J. Huang, and G. Qiu. Jpeg error analysis and its applications to digital image forensics. *Information Forensics and Security, IEEE Transactions on*, 5(3):480 –491, 2010.

[17] A. Popescu and H. Farid. Exposing digital forgeries in color filter array interpolated images. *Signal Processing, IEEE Transactions on*, 53(10):3948–3959, 2005.

[18] A. Popescu and H. Farid. Statistical tools for digital forensics. In *Information Hiding*, pages 395–407. Springer, 2005.

[19] H. Sencar and N. Memon. Overview of state-of-the-art in digital image forensics. *Algorithms, Architectures and Information Systems Security*, pages 325–344, 2008.

[20] A. Swaminathan, M. Wu, and K. Liu. Nonintrusive component forensics of visual sensors using output images. *Information Forensics and Security, IEEE Transactions on*, 2(1):91–106, 2007.

[21] A. Swaminathan, M. Wu, and K. Liu. Digital image forensics via intrinsic fingerprints. *Information Forensics and Security, IEEE Transactions on*, 3(1):101 –117, 2008.

[22] G. K. Wallace. The jpeg still picture compression standard. *Commun. ACM*, 34:30–44, April 1991.

Dynamic Formation and Opponent Modeling in Real Time Strategy Games

Amirhosein Shantia

Abstract—Real Time Strategy Games are controlling the highest share in PC markets. They present an environment in which players can control a base and a set of units in real time. The main steps of playing are unit micro management, building order, resource management, and the game main tactic. Unfortunately, current games mainly use fixed or limited dynamic scripting to control the non-human player, and therefore, the player can easily learn the counter measures to defeat the AI. Recently, research has been done on modeling the main behavior of the opponent in order to execute counter tactics and win the game. However, the experiments were done on non/weak-commercial open source games. In this paper, we review the most popular opponent modeling methods and will implement these methods in the popular StarCraft game. The goal is to predict the opponent's action and select a reasonable counter measure.

Index Terms—Opponent Modeling, Dynamic Formation, RTS, Neural Networks, Reinforcement Learning.

1 Introduction

Real-time strategy games(RTSG) are a sub-class of strategy video games which as it name says is real time and not turn based. In an RTSG, the player has control over a base, a set of units and structures with the goal to secure areas of the map and/or destroy other non-friendly players' bases. In a typical RTSG, the creation of unit and building constructions are generally limited by a requirement to have special resources. These resources can be gathered by controlling special points on the map and/or possessing certain types of units and structures devoted to this purpose. More specifically, the main steps of playing are unit micro management, building order, resource management, and the game main tactic. StarCraft, WarCraft, Age of Empires, and Command & Conquer are examples of such RTS games[3].

Gameplay in RTSG generally consists of the players being randomly positioned somewhere on the map with a few units or a building that is capable of building other units/buildings. Usually, the player must construct a set of structures to unlock more advanced units or structures in the tech tree. In addition, in all RTS games, the player must build an army to either defend him/herself from a virtual form of attack or to completely eliminate enemy structures. In order to achieve this, the player should keep a strong economy by Resource gathering. Other titles of the genre place higher gameplay significance to how the units are used in combat. Some titles impose a ceiling on the number of simultaneous troops, which becomes a key gameplay consideration, a significant example being StarCraft, while other titles have no such unit cap.

Another important factor to win the game is to choose a correct tactic against the opponent. For instance, if one knows what types of structures the opponent has, then typically one would understand the mixture of the opponent units and choose to build units that are strong against those of the opponent. To make predictions about the opponent's strategy, an AI player can also establish an opponent model. Many studies confirm the importance of opponent modeling and predicting opponents actions, such as, [12][1][9][3][14][17]. In these studies, it is also stated that opponent models are required to deal with the complexities of state-of-the-art video games [16]. One of the biggest issues in opponent modeling in RTSG is that the player does not have access to all the information and its knowledge is limited to the visibility range of units and structures. This makes the construction of opponent models in an RTS game a difficult task.

Opponent Modeling is not limited to RTS games. Extensive research have been done by researchers for other games, such as poker, backgammon, scrabble, etc. Baker, *et al.*, used Bayesian opponent modeling. Opponents were defined in four distinctive styles, and tactics were developed which defeat each of the respective styles [2]. Jozi *et al.*, they proposed a behavior structure for agents which consisted of low level skills and high level skills , and also a simple method of opponent modeling called harmonic opponent modeling which model the walking speed of opponent agent [11]. Billings *et al.*, described and evaluated Loki, a poker program capable of observing its opponents, constructing opponent models and dynamically adapting its play to exploit patterns in the opponents play [4]. Schad *et al.*, proposed a method for modeling opponent in Spring game. However, their focus were mainly on the units of the opponent. Typically, when the units are produced, it is already too late to start a counter measure. The prediction should be done by also taking into account the type and of number buildings and workers of the opponent [14].

In addition, Real time strategy games are not only an interesting topic for sole research purposes. The video game industry has roughly gained eleven billion dollars by selling games in 2008 and 2009. The share for PC games are around seven hundred million dollars. Real time strategy games dominate the PC games sell by 35.5%. Therefore, any breakthrough in this field is a potential investment for financial success [7].

Contribution. In this paper I explore whether opponent modeling in the game tactic can be useful for obtaining good performing game AI for selecting the main tactic of the match in the game StarCraft. First of all, I introduce the subject of opponent modeling. Next, I describe my approach and implementation of opponent modeling in StarCraft which is the only popular and commercial real time strategy game that is possible to manipulate.

Outline. In the next section, first, I describe the StarCraft game. In Section III, I present bases of opponent modeling and my proposed modeling approach and implementation. In Section IV, I conclude this paper.

2 StarCraft ™

StarCraft ™ is a popular Real-Time Strategy (RTS) video game developed by Blizzard Entertainment in 1998. StarCraft is the most successful RTS game with more than 9.5 million copies sold as its released date in 1998 until 2004. StarCraft consists of three races (Terran, Protoss, and Zerg), each of which have special building structures and different types of soldiers to engage in battle. The game-play in these games involves micro and macro management. Micro management involves focusing on unit control while macro management focuses on building order, resource management, and the game main tactic. In this paper I focus on macro-management of the game by trying to understand the opponents tactic. The game has a terrain with

• *Amirhosein Shantia is with Department of Computer Science, University of Groningen in The Netherlands, E-mail: a.shantia@student.rug.nl.*

Fig. 1. Battlefield and interface of a StarCraft Match.

different possible heights, choke points, etc. Each unit of the game has a number of attributes such as health, armor, weapon, move speed, attack speed, turn speed, etc. Player unit information is always accessible to the player, but enemy unit information can be chosen to be visible or hidden when it is out of sight. Figure.1 shows the interface and battlefield of the StarCraft Game. In this paper, in order to reduce the possible tactics and counter measures between different races, I mainly focus on opponent modeling for Terran vs. Terran matches.

3 Opponent Modeling

In general, an opponent model is a short description of players' behaviors in a game [16].Opponent modeling can be counted as a classification problem, where the input data is usually the type of structures, units, and attacking time of enemies, and the output labels regarding to the input data will be the type of opponents strategy. A limiting condition is the fact that in RTS games, these classifications have to be performed in real-time, while many other computations, such as rendering the game graphics, have to be performed in parallel. However, with the advent of multi-core technology, the computation speed of computers increased significantly. Still, computationally-inexpensive techniques are preferred for opponent modeling in RTS games. Preference-based modeling is a commonly used computationally light technique [5]. The method identifies the model of an opponent by analyzing the opponent's choices in important game states, such as building orders, unit composition and etc. Due to the visibility limitations in RTS games, however, it is common that choices of the opponent cannot always be observed and scouting is required to increase the precision of the model.

The approach I use in this paper is the following. First, a set of known tactics are selected as classes. Then, based on opponents structure and unit building order, a number of possible opponent models is established. Next, the confidence level of each opponent model is calculated, and finally, the opponent model with the highest confidence level is selected. Since the state space of the game is huge and complex, a better approach is to apply a hierarchical ordering on the possible opponent models [10]. This hierarchical approach allows us to divide the complex modeling in different simple problems . In addition, the hierarchical approach makes it feasible to use different classification methods in each level of the hierarchy. For establishing opponent modeling in RTS games, we follow the hierarchical method. As previously said, the most defining element of an opponent's strategy is the main tactic chose by the player. We therefore place the general play style at the top of the hierarchy. Each play style has its own sub-tactics that further demonstrates behavioral characteristics. For example, if it is known that the opponent is aggressive, a logical response would be to improve one's defenses. If also the opponent's choice of units is known, the defenses can be specialized to be effective against those specific units.

3.1 Implementation

This section discusses my implementation of hierarchical opponent modeling in StarCraft for the Terran Race. In my implementation, I start establishing a hierarchy consisting of three levels. The first-level of the hierarchy classifies the opponent's general play style. The second-level of the hierarchy classifies the opponent's choice of building structures, and finally, the third-level of the hierarchy classifies the opponent's choice of building units. In StarCraft, we discriminate between an aggressive, and a defensive play style. For an aggressive play style we discriminate at the third level between pre-dominantly using the following four unit-types: (1) Siege Tanks, (2) Marines, (3) Vultures(Fast Bikes), and (4) Airplanes. Each unit-type has specific strengths and weaknesses, and is used to execute a particular strategy. For instance, Vultures are relatively fragile but are very fast and have long shooting range, and are therefore useful for a strategy against an opponent which attempts to attack with small and slow units. Tanks can only maneuver on plain terrain but are relatively sturdy, and are therefore useful for a strategy against an opponent who constructs strong defenses. Also, tanks in siege mode can cover a larger area and do significant amount of damage. For a defensive play style we discriminate at the bottom level between the following three building preferences: (1) Super Weapon, (2) Tech, and (3) Bunker. These three building preferences are commonly observed in actual StarCraft games. The mentioned hierarchy defines the following strategies:

1. Aggressive → Vultures. The opponent will attack early to disrupt resource gathering units and damage economy.
2. Aggressive → Marines. (Large Number of Troops). The opponent will do an all-in attack in order to defeat the enemy.
3. Aggressive → Siege Tank. The opponent will damage the front line by use of the siege mode.
4. Aggressive → Airplane. The opponent will damage the resource gathering units to damage economy.
5. Defensive → Bunker. The opponent will make bunkers near its base choke point in order to repel any possible attacks.
6. Defensive → Tech. The opponent will stay inside its base until it makes high end units.
7. Defensive → Super Weapon

The information about the labels are based on years of human expert play style [13],[6].

3.2 First-Level Classifier

In this hierarchy, the top level classifier is used to distinguish between 'aggressive' and 'defensive' opponents. An aggressive player typically will spend a large part of game time attacking. A defensive player, on the other hand, typically, will use most of the game time for preparing an army (with high end units), and only needs a small amount of the game time for an actual attack. As a result, a rational feature to discriminate between these two classes is a way to measure attack time of the opponent. An attack is defined as the time when the AI agent loses a number of its own units. An example of a top-level player model is shown in Figure 2. which is the calculated confidence rating in the SPRING game by [14]. The Figure illustrates the confidence of the opponent following an aggressive and defensive play style, as a function of the percentage of game time spent on attacking. The data were gathered from several manual games of a good human player.

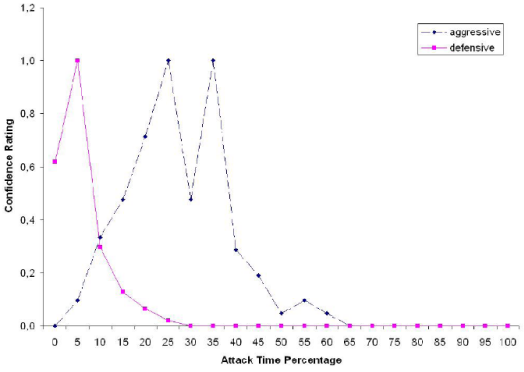


Fig. 2. Example of a First-level player model.

3.3 Second and Third-Level Classifier

After detecting game style, I need to find out which tactic the opponent is going to use against the player. The second and third-level classifier has to discriminate between the sub-tactics that further demonstrates behavioral characteristics of the opponent. However, during the game, a player can switch from one tactic to another or use a mixture of tactics. Therefore, the classifier needs to be able to map the new data to one of the pre-defined tactics. It also requires to emphasize recent events of the match more than past events. To achieve this, I use a reward function for the system in order to determine whether it correctly anticipated the opponent's tactic. The reward can be selected as the match result or the army value of each team after a single battle during a match. Since the goal is to adapt with the opponent's tactic over time, we have to also introduce a discount factor to decay the results that came from past. Over time, if a deviation from the selected tactic is required, the tactic which gave higher rewards in the past experiences will be chosen. The expected reward formula can be computed as follows[8]:

$$(1-\delta)\sum_{1}^{\infty}\pi_i \times \delta^{i-1} \tag{1}$$

Here the discount factor δ and reward at time t π_t is applied. Since it is assumed that a player prefers to receive a reward as soon as possible, rewards in the future are valued less. This is based on Occam's Razor concept in which it is unlikely to find a short correct path by coincidence. On the other hand, finding a long path that has a much higher probability to be a coincidence. This valuation is expressed by multiplying the future reward with the discount factor. When the expected rewards are calculated, a simple selector mechanism usually will select the strategy with the highest expected reward [15].

Modification for RTS:
Similar to calculating the expected rewards, for discriminating between opponent strategies, I also require to measure the confidence rate of the opponent applying a particular strategy. The confidence value of the anticipated opponent's strategy is calculated during the game. In classical matrix games, a game event would be one move of both players. However, because the nature of RTS is real-time, the term of an event must be rephrased. As stated before, the occurrence of an attack is a suitable event to detect whether the player is aggressive, and to calculate the confidence level of the player decision. When playing against a defensive opponent, however, waiting for an attack of is typically an unsuccessful game strategy and will usually result in a loss. Against a defensive opponent, it is crucial that the player 'scout' the base of the opponent. Scouting will provide information about the opponent's actions. When an event is detected, the confidence values of each possible strategy will be updated according to the observed game information. If δ is the discount factor, $\psi_{s,t}$ the belief that the opponent uses strategy s at event t, ranging between 0 and 1, and π_t the total reward added at each event and i the most recent event, then the confidence rate c_s that the opponent uses strategy s is computed as follows in formula 2 :

$$c_s = \sum_{t=i}^{0} \psi_{s,t} \times \pi_i \times \delta^{i-1} \tag{2}$$

The belief parameter $\psi_{s,t}$ is calculated by observing all possible enemy info. Each unit or structure has a value representing a tendency to a certain strategy. The unit-tendency values were determined by the supervised human expert, using his own knowledge of the game. To give three examples of unit-tendency values: (1) a number of enemy bunkers has a high tendency towards an opponent using the defensive bunkering strategy, (2) two Barracks early in game has a high tendency towards an aggressive opponent (3) a siege tank has a tendency towards both the aggressive and defensive strategy.

4 Conclusion

In this paper, I described opponent modeling and its importance in game AI.I introduced my approach for implementing opponent modeling in the real time strategy game, StarCraft. StarCraft is one of the most famous real time strategy games developed until now, and has recently been used in tournaments to compare the use of different techniques from artificial intelligence.

I used a hierarchical opponent modeling in order to break down the complex problem of modeling into separate simpler problems. The final architecture consists of three modeling levels, namely, Aggression , Structure and unit level modeling. The top-level of the hierarchy can classify the general play style of the opponent (Aggression). The bottom-level (structure and unit) of the hierarchy can classify strategies that further define behavioral characteristics of the opponent.

References

[1] B. Abramson. Expected-outcome: a general model of static evaluation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(2):182–193, Feb. 1990.
[2] R. Baker and P. Cowling. Bayesian opponent modeling in a simple poker environment. In *Computational Intelligence and Games, 2007. CIG 2007. IEEE Symposium on*, pages 125–131, april 2007.
[3] S. C. Bakkes, P. H. Spronck, and H. J. van den Herik. Opponent modelling for case-based adaptive game AI. *Entertainment Computing*, 1(1):27–37, 2009.
[4] D. Billings, J. Schaeffer, and D. Szafron. Opponent modeling in poker. pages 493–499. AAAI Press, 1998.
[5] J. Donkers and P. Spronck. *Preference based player modeling*. Charels River Media, Boston, Massachusetts, 2006.
[6] B. Entertainment. StarCraft Brood War. http://us.blizzard.com/en-us/games/sc, 2011. [Online].
[7] ESA. Essential facts about the computer and video game industry, 2010.
[8] R. Gibbons. *A Primer in Game Theory, Chapter 2*. Pearson Higher Education, June 1992.
[9] R. Gibbons. *A Primer in Game Theory, Chapter 2.3B*. Pearson Higher Education, June 1992.
[10] R. Houlete. *Player modeling for adaptive games*. Charels River Media, Hingham, Massachusetts, 2004.
[11] B. Jozi, A. Fakharian, M. Nademi, and M. Khanian. Harmonic opponent modeling and behavior structure for 3d soccer simulation agent. In *Computational Intelligence in Robotics and Automation (CIRA), 2009 IEEE International Symposium on*, pages 394–397, dec. 2009.
[12] R. E. Korf. Generalized game trees. In *Proceedings of the 11th international joint conference on Artificial intelligence - Volume 1*, pages 328–333, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.
[13] T. Liquid. StarCraft Wikipedia. http://wiki.teamliquid.net/starcraft/Main_Page, 2011. [Online].
[14] F. Schadd, S. Bakkes, and P. Spronck. Opponent modeling in real-time strategy games. pages 61–68, 2007.
[15] F. Schaeffer. *Hierarchical opponent models for real-time strategy games*. Universiteit Maastricht, 2007.

[16] H. J. van den Herik, H. H. L. M. Donkers, and P. H. M. Spronck. Opponent modelling and commercial games. In G. Kendall and S. Lucas, editors, *Proceedings of IEEE 2005 Symposium on Computational Intelligence and Games CIG'05*, pages 15–25, 2005.

[17] M. van der Heijden, S. Bakkes, and P. Spronck. Dynamic formations in real-time strategy games. In *Computational Intelligence and Games, 2008. CIG '08. IEEE Symposium On*, pages 47–54, december 2008.

Cloth Simulation: Permanent Deformation using Hysteresis

Dirk Zittersteyn Tijmen Klein

Abstract—Cloth simulation using springs and dampeners has been a subject of research for over a decade. Many simulations take inspiration from molecular dynamics, modelling cloth as a grid of nodes interconnected by springs. These springs do not support permanent deformation, so cloths will always return to their original shape. This creates some unrealistic effects when large external forces are applied, such as a flag during gale force winds or the firing of a bullet at a Kevlar vest. We extend this simulation to create more realistic and versatile results.
We propose a method that uses per-spring hysteresis to make a better approximation of a natural spring. A system shows hysteresis when its behaviour depends not only on the current input parameters, but also on the input it received in the past. When a force that causes the spring to exceed its yield point is applied, the state of the spring is permanently changed. The spring loses some of its stored energy in the process of deformation, meaning it does not contract to its original length and the force exerted at a certain length is reduced. We simulate this deformation by recording the stresses the spring has gone through.

Index Terms—Cloth simulation, hysteresis, springs, molecular dynamics, computational science.

1 Introduction

Cloth simulation is a subject that presents several different methods. Three main methods can be distinguished: [9]

- geometric,
- particle/energy, and
- physical.

Geometric methods, first used by Weil [11] in 1986, use a model based on cable tension. Weil states that due to the high computational complexity of this method, live computation is not feasible.

Particle/energy methods are based on molecular dynamics-approaches for simulating interaction between particles. The cloth is modelled as a collection of nodes, each interacting with all other nodes based on simple rules of attraction and repulsion.

The particle/energy methods are still quite computationally complex, as every particle has to interact with all other particles in the cloth. For a cloth with n nodes this will be $O(n^2)$ computations. Using a neighbourhood list we can reduce the computational complexity to $O(c \cdot n)$, where c is the size of the neighbourhood list. When using a neighbourhood list with a small size, inaccuracies will occur, for example in sharp curves. In this case nodes may become detached from the cloth.

A way of preventing this is by using the physical method, where the neighbourhood list is replaced by a fixed number of adjacent nodes, which are connected to the node by springs as in Breen *et al.*[2]. Breen *et al.* state that realistic results can be achieved with only 12 springs, connected as in Figure 1. Due to the realism achieved at low computational costs (compared to geometric and particle/energy), we use this method as a basis for our work.

The work of Breen[3] presents a physical-based model to predict the drape of different types of woven fabrics. Photographs of real fabrics are used as a reference for realism. The Kawabata Evaluation System [5] is used to determine the parameters of the fabrics. Kawabata uses several testing setups to determine the properties of real fabrics, these can then be used in simualations. Another approach uses a genetic algorithm to extract the parameters from a video showing the fabric[1]. This way any fabric can be simulated using only a video of the fabric in motion as an input.

Most implementations of the physical method use Hooke's model to compute the force a spring exerts. Hooke's model is realistic for behaviour at small amounts of stretching. Unrealistic behaviour surfaces when the limits of the system are explored. The cloth will always return to its original state, no matter how much it was stretched, causing unrealistic physics. To improve the realism of this method we look into the behaviour of springs at their limits.

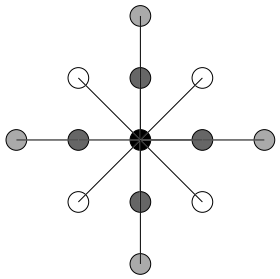


Fig. 1. The black node is connected to structural (dark), bending (light) and shearing (white) points.

In this work we show our implementation of realistic springs, as well as the results it yields with regard to cloth simulation. We go into our method for simulating cloth as well as the adaptation we made to account for permanent deformation. An example of a cloth that shows permanent deformation is shown in Figure 2. This cloth was hit by a sphere in the upper right corner.

For cloth simulation we use the molecular dynamics based framework of Moosegaard[7], which we extend to conform more closely to the natural behaviour of springs.

2 Method

Our cloth simulation is based on a physical method, where the nodes are connected with springs. The simulation of these springs determines the new positions of the nodes.

2.1 Physical method

A cloth simulation based on the physical method consists of repeating the following basic steps:
(2.4.1) calculate forces applied by the springs,
(2.4.2) determine the resulting acceleration of the nodes,
(2.4.3) calculate the velocity of the nodes,
(2.4.4) move the nodes based on these velocities, and
(2.4.5) increase the current time by the time step.
Springs form the connections between the nodes, and we can distinguish three different types of springs [10], which are explained below. How these springs are connected can be observed in Figure 3.

- *Dirk Zittersteyn @ University of Groningen*
  E-mail: D.L.Zittersteyn@student.rug.nl.
- *Tijmen Klein @ University of Groningen*
  E-mail: T.R.Klein@student.rug.nl.

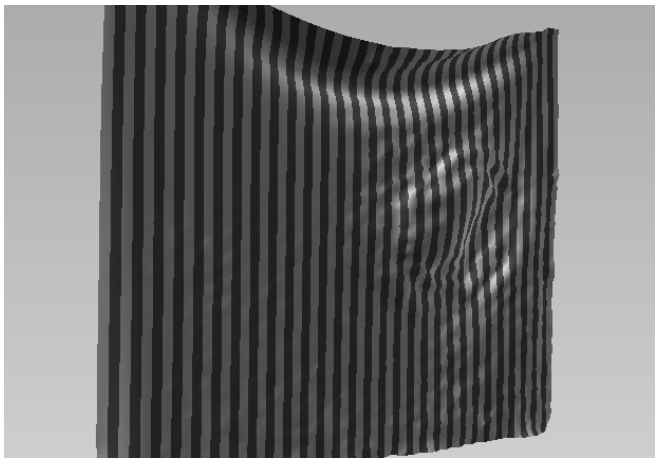


Fig. 2. A piece of cloth showing permanent deformation.

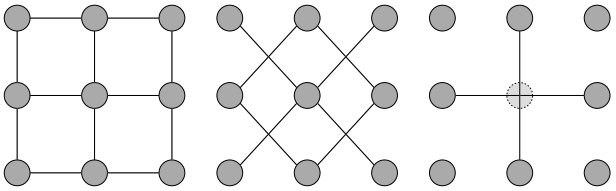


Fig. 3. The different types of springs used.

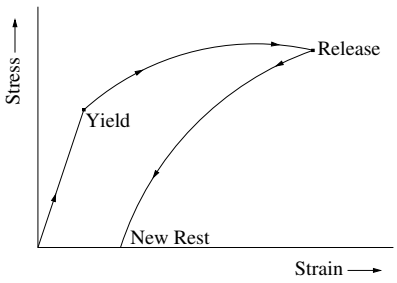


Fig. 4. Stress vs. strain graph of a permanently deforming spring.

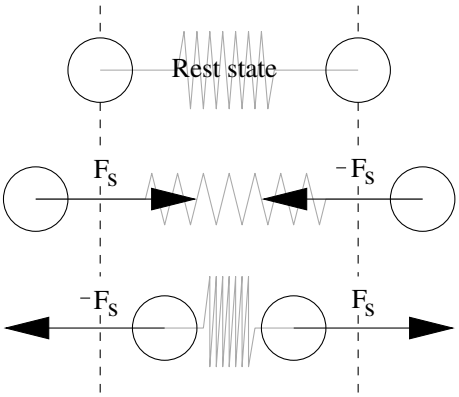


Fig. 5. Forces exerted on two nodes that are connected with a spring.

2.1.1 Structural springs

Structural springs are the springs that define the main structure of the cloth, and connect every node with its 2 vertical and 2 horizontal neighbours. Structural springs simulate the actual fibers in the cloth. The strength of these springs determines the load bearing properties of the cloth.

2.1.2 Shear springs

Shear springs connect each node with the 4 diagonal neighbouring nodes. These springs restrain shear deformations. Shear springs simulate the force two adjacent fibers exert on each other. A loosely woven piece of cloth will have very low shearing recovery, while very dense cloth will exert a lot of force to return to its square resting state.

2.1.3 Bend springs

Bend springs restrain the bending of the material. A piece of satin for example has very low strength bending springs; a piece of rubber on the other hand has very high bending recovery.

2.2 Springs

Real-life spring deformation can be modelled using Hooke's law [4], which states that the spring force is a product of the difference between the rest length and the current length (δx) and the spring constant (k), so:

$$F_s = \delta x \cdot k \tag{1}$$

As long as the force that is applied on the spring stays below its yield point, this is an accurate approximation. However, when the amount of deformation exceeds the yield point Hooke's law does not hold anymore. In the stress vs. strain graph in Figure 4 the relation between the stress and the strain on the spring is shown. The line segment from the origin to the yield point shows a linear relation between the two, after the yield point however this relation becomes irregular. The yield point denotes the amount of stress a spring can take before its molecular structure is permanently changed. This deformation reduces the amount of potential energy in the spring, resulting in a changed rest-state. The analogy holds for our cloth simulation, as a piece of cloth will return to its original state after a small force is exerted, but will permanently deform after a large amount of stretching. To approximate this permanent deformation, an extension to Equation 1 is required.

2.3 Deformation and Hysteresis

The behaviour of a spring when stretched beyond its yielding point is quite complex. As can be seen in Figure 4, the amount of deformation a spring undergoes depends on how far the spring exceeds its yielding point. Using this approach will most likely result in computationally heavy simulation. Therefore, we choose to model the spring in a simpler way. We assume a spring will never break, and will not change structure when stretched. This means a spring will always keep the same spring constant k. When a spring exceeds its yielding point we simply extend the spring by the amount by which the yield point was exceeded. This amount is called h. This results in the simulation shown in Figure 6 and 7.

We implement this using a method very similar to hysteresis. A system that exhibits hysteresis (also called 'path-dependance') will react differently based on how it has reacted in the past. An example of hysteresis can be seen in automatic street lights. An automatic street light will turn on when the light level is below a certain threshold T, and off when the light level is above T. Without hysteresis this system can show rapid on-off flickering when the light level is close to T. To prevent this, a street light will only turn on when the light level is below $T - \alpha$, and only turn off when the light level is above $T + \alpha$.

We apply a similar method to our springs, as we take previous stretching into account. Every spring keeps track of its own length, and changes this length according to how much the yield point has been exceeded.

2.4 Simulation

Simulating a continuous process like cloth deformation would require us to analytically solve several differential equations. Another way of

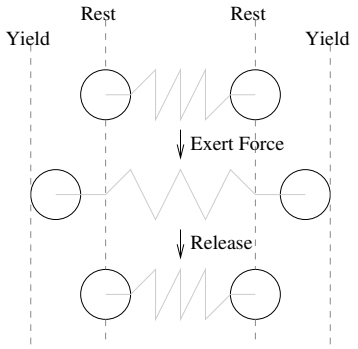


Fig. 6. Stretching a spring within its limits. The spring (eventually) returns to its initial rest state.

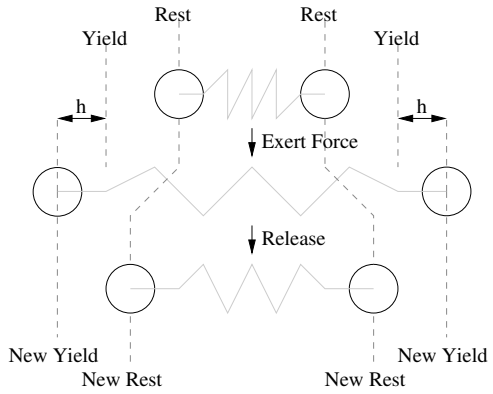


Fig. 7. Stretching a spring beyond its limits. Permanent deformation occurs, changing the rest state and yield point of the spring.

approaching this problem is to discretize time into a large number of time steps, and use the finite difference method to discretize the cloth. This way we can use a numerical approximation of the movement of the cloth, which results in a simpler model.

The steps described in 2.1 are applied each time step.

2.4.1 Calculate forces

First we compute the forces exerted by the springs connected to each node. This is done by first computing $k \cdot \delta x$, where δx is based on both original and current length, L_o and L_c respectively, and a hysteresis constant h:

$$\delta x = L_c - (L_o + h) \tag{2}$$

In this formula $L_o + h$ represents the current rest distance.

We then use Equation 1 to compute the force exerted. When combined with Equation 2 this formula is:

$$F_s = k \cdot (L_c - (L_o + h)) \tag{3}$$

In Figure 8 a plot of stress against strain for the model we use can be seen. While the model is not accurate (compare Figure 4), the important points (Origin, Yield, Release and New Rest) correlate. This means the behaviour is similar.

2.4.2 Determine acceleration

Using Newton's second law [8], combined with the known mass of a particle and the forces that act on it, we can compute the acceleration the particle will undergo. Newton's second law is stated as: $F = m \cdot a$. This can be rewritten to:

$$a = \frac{F}{m} \tag{4}$$

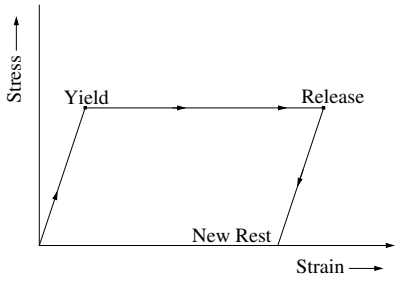


Fig. 8. The approximation of the stress-strain model used in our model. Also see Figure 4.

2.4.3 Calculate velocity

To calculate the velocity of a particle, we integrate a over t (time), so $\int a(t)\,dt$. We discretize this according to the midpoint rule on an equidistant grid. We use the approximation

$$\int_x^y a(t)\,dt \approx \sum_{i=1}^{n} a(t + (i-1)\Delta)\Delta \tag{5}$$

Equation 5 approximates the area under a curve as a series of n rectangles (this is the midpoint rule). The time step is denoted as Δ.

2.4.4 Move the nodes

It is easy to see that $s = \int v(t)\,dt$. We use the same approximation of this integral as Equation 5 to calculate how far the node will move.

2.4.5 Increase current time

The choice of the time step is critical, as a time step that is too small will result in a slow simulation, while a too large time step will cause instability in the integration of both a and v. Meyer[6] shows that the maximum time step is inversely proportional to the square root of the spring constant of the material, a property dicussed in Section 3.2.

3 Implementation

To implement the hysteresis that is described in Section 2, we need a standard physics-based cloth simulation to start with. This basic cloth simulation program is extended to implement per-spring hysteresis. This section describes the framework that is used as the starting point and explains the hysteresis-extension that is applied to this framework.

3.1 Basic cloth simulation framework

We use the C++ based Cloth Simulation Coding Tutorial from Mosegaard[7] as the starting point for our cloth simulation. The tutorial is used to explain the basics of a physics based cloth simulation and has minimal external dependencies. This framework supplies the data structures for repulsion and attraction, as well as nodes. It also implements some basic vector mathematics.

Furthermore the framework provides an initialization that properly connects the nodes with its neighbours using structural, shearing and bending constraints. An example of how the immediate neighbours are connected can be seen in Figure 9. However, these constraints do not act as springs, as they do not follow Hooke's law and therefore are not suitable for hysteresis. As a nice side-effect, a spring-based calculation is faster than a constraint based method. Based on these reasons we replace the constraint class with a spring class.

The cloth consists of a grid of nodes, and can be visualised by drawing 2 triangles between 4 nodes of the cloth. The framework takes care of this visualisation and also calculates the normal vectors for every triangle, which helps to create more realistic lighting. The function `drawTriangle()` draws a triangle based on the the vertices and normals of 3 nodes, as shown in Figure 10.

The framework has a method that shoots a ball into the simulated piece of cloth. This can be used to test the deformations of the cloth. However, the approximation of the collision that is used for this is

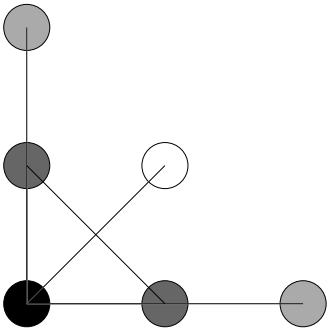


Fig. 9. Connecting a node with the appropriate neighbours.

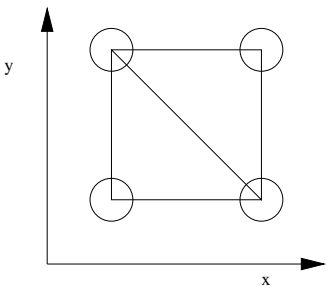


Fig. 10. The triangles that are drawn

rather simplistic, and can give unrealistic and unwanted results. The ball is simply moved a certain distance every timestep. When a node ends up inside the ball, the node is moved to the nearest point outside the ball. This means the ball has an infinite mass, which can cause extreme deformation in some cases. Another downside of the current implementation of the ball is the fact that the ball moves a fixed distance every time step. This means that impact of the ball depends on the size of the time step.

3.2 Spring class with hysteresis

One time step for the spring and nodes consists of two basic steps: performing the spring calculations and moving the nodes. The spring calculations are executed first, since these exert forces on the nodes, which gives the nodes their speed. These discrete calculations depend on the global time step value that is defined in the program.

The calculations of our spring class are primarily based on Equation 1 that has been extended with hysteresis as described in subsection 2.3. All the springs have a spring constant k, a rest distance L_o that is based on the initial distance between the two nodes, a constant yield factor y, and a hysteresis value h, that are used for the spring calculations. The function `satisfySpring()` (Listing 1) applies Hooke's law with hysteresis and adds the resulting forces to the 2 nodes that are connected to the spring. The hysteresis value is adjusted if the current displacement (δx) is larger than the current yield point. The yield point can be derived from y and the current rest length.

To summarize: a spring is defined as:

- k: the spring constant,
- L_o: the initial rest distance,
- y: the yield factor,
- h: the hysteresis value and,
- δx: the current amount of displacement

To evaluate our method we first implemented a simple simulation of a 1D spring suspended at a certain height, as shown in Figure 11. The results of this simulation are plotted in Figure 12 and 13. In the first figure the spring starts oscillating around a single point, with the top of its oscillation level with the original height. In the second figure hysteresis is introduced, causing the spring to permanently deform during the initial drop. As the spring stretches, the yield point is placed

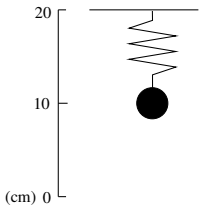


Fig. 11. The simulated model of a one-dimensional spring.

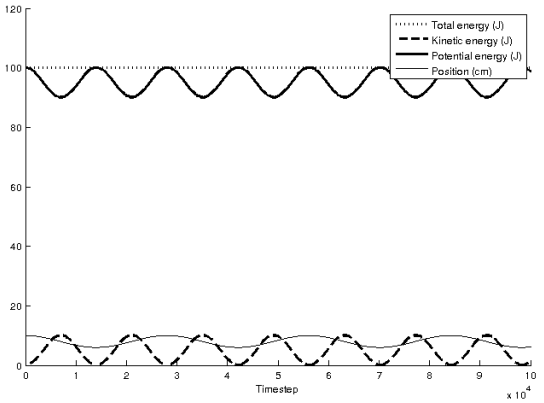


Fig. 12. Simulation of the spring in Figure 11, without hysteresis. It can clearly be seen that the total energy in the system remains constant.

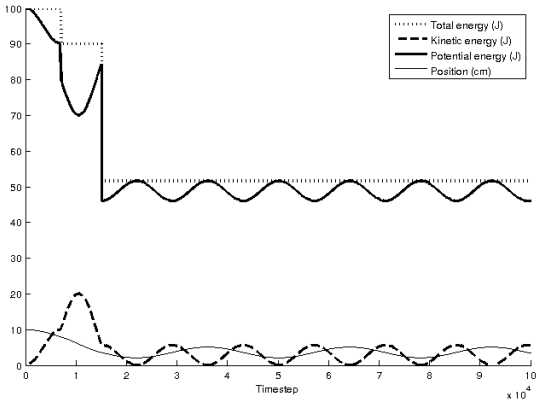


Fig. 13. Simulation of the spring in Figure 11, with hysteresis. A couple of discontinuities can be seen, corresponding with the deformation of the spring. The energy that seems 'lost' is converted to heat.

further and further (since the yield point is based on y and the current rest length), so eventually the fall is stopped and the spring will start oscillating around a single point.

The parameters of the springs determine the characteristics of the cloth, changing the parameters can give different kind of materials. Therefore it is convenient to give these parameters as run time arguments to the program. The following parameters can be giving as arguments:

- damping factor,
- spring contact,
- yield factor (that determines the yield point),
- step size of the time,
- gravitational constant.

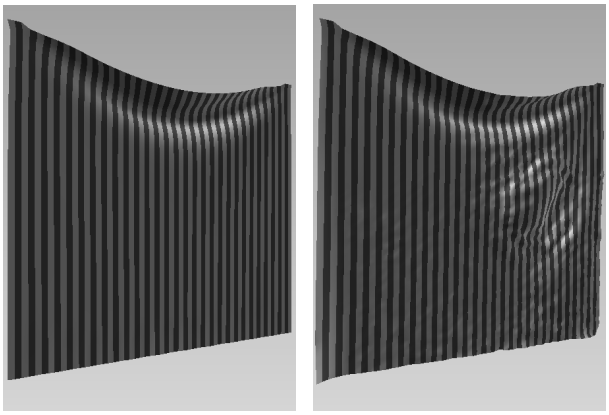
Listing 1. Spring calculations

```
void satisfySpring()
{
  // Vector from node 1 to node 2
  Vec3 p1_to_p2 = p2->getPos() - p1->getPos();
  // Distance between the nodes
  float current_distance = p1_to_p2.length();

  // Hysteresis
  if(current_distance - rest_distance - h > yield){
    h = current_distance - rest_distance - h - yield;
    yield = (rest_distance + h) * YIELD_FACTOR;
  }

  // The correction vector to move the 2 nodes.
  Vec3 force = p1_to_p2.normalized() * (current_distance -
       rest_distance - h) * SPRING_CONSTANT * 0.5 ;

  // Apply the forces
  p1->addForce( force);
  p2->addForce(-force);
}
```



(a) Without permanent deformation (b) With permanent deformation

Fig. 14. A piece of cloth after collision with a sphere.

3.3 Simulation

Our simulation creates a piece of cloth that is defined on a 60×60 grid of nodes. The nodes of the top corners of the cloth are fixed, which results in the cloth hanging from these 2 points because of the gravitational force. It is also possible to create a cloth with one, three of four fixed corner points.

It is hard to test the permanent deformation of the cloth with the shooting of the ball. Therefore, another method to deform the cloth is added: pulling. For a certain amount of time a constant force along the Z-axis is exerted to a node in the centre of the cloth (the cloth is initally placed along the XY-plane). This results in a realistic pull that does not depend on the time step, which means that the pull method can easily be used to test the permanent deformation of different kind of materials.

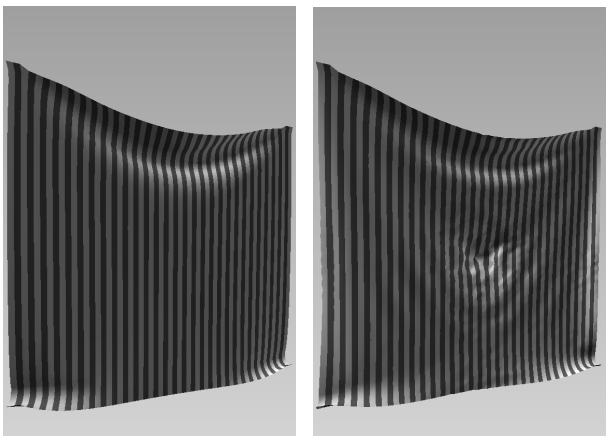
4 Results

The results that are obtained by extending the spring simulation with hysteresis look realistic. This section shows and discusses some of the obtained results. All test simulations were done using consumer hardware and are based on a 60×60 grid of connected nodes. The simulation runs in real-time on this setup, and is able to top 250 frames per second on current consumer hardware.

Figure 14 clearly shows the difference between a simulation without and with permanent deformation. The cloth in Figure 14(a) shows a deformation on the location where the collision with the sphere occurred. The springs at this location have undergone such strong forces that they were stretched beyond their yield point, and thus their rest length is increased by means of permanent deformation. This increased rest length is easily observable in Figure 14(b), and accounts for the non-uniform look.

Figure 15 shows the result of the pulling force on the cloth. The cloth shows nice permanent deformation at the location where the force is applied. The deformation is severe at the node that is actually pulled, and quickly decreases around this point. The deformed area is relatively small when compared to the deformations of the impact with a sphere (Figure 14(b)); which is expected since the sphere interacts with more nodes.

The time step that is used can not be too large, as is explained in Section 2.4.5. A time step that is too large will quickly result in very unstable simulations. An example of such an unstable situation is shown in Figure 16.



(a) Without permanent deformation (b) With permanent deformation

Fig. 15. A piece of cloth after applying a pulling force at the center.

Fig. 16. Large time step causing instability

5 Conclusion & Discussion

Hysteresis is a valuable addition to spring based cloth simulations. The calculations that are required to add the hysteresis do not give much overhead, which means that the simulations can still run in real-time. The overall results are convincing and the visualisations of the permanent deformations look realistic. The simulation currently is useful for deformation of for example kevlar in bulletproof vests, and for illustrative purposes.

However, hysteresis does not always increase the realism of a simulation. Most simulations will not exert such large forces on the springs that they should be permanently deformed, which means that the hysteresis calculations do not influence the simulation. On the other hand, no material will keep on stretching forever without tearing, which is a parameter that is not taken into account in our method.. The cloth will never tear, even when extreme forces are applied to it, therefore the method is unsuitable for simulations where the interest lies in the breaking points of materials.

This method can also be used for other situations where physical methods are used, such as solid media deformation. This would require a 3D grid of nodes, and springs in more directions. The simulation of the springs and nodes can remain the same as in our simulation. Real-world applications could be car crash simulations, or collisions of stellar bodies. A simple example of a physical method in 3D is the simulation of a piece of clay.

The current implementation still has some defects. The spring constant of all springs is the same. We would like to make the structural and shearing constraints stronger than the bending constraints, to make it possible to make very flexible, heavy materials that can hold up their own weight.

Further problems occur in intersection calculation are caused by the fact that we don't check for self-intersection. This however is a problem of the physical method, and is not easily amended.

Another oddity is that the cloth is perfectly flat when the simulation starts, causing a unstable equilibrium that causes the fabric to stand upright when fixated at the bottom two corners. This is solvable by introducing a small perturbance, breaking the equilibrium. After disturbing the cloth it will collapse into a more realistic state. More realistic would be to introduce a small disturbance in the force-computations, collapsing this state immedeately.

The current implementation is not very suitable for porting to the graphics card (GPU), as it contains some if/else branches. However, these branches can be eliminated with the use of min/max functions. Since GPU's currently far exceed the capability of CPU's in Single Instruction, Multiple Data (SIMD) situations, porting to GPU could be very interesting.

Acknowledgements

The authors wish to thank dr. H. Bekker, Assistant professor at the University of Groningen.

References

[1] K. Bhat, C. Twigg, J. Hodgins, P. Khosla, Z. Popovic, and S. Seitz. Estimating cloth simulation parameters from video. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 37–51. Citeseer, 2003.

[2] D. E. Breen, D. H. House, and P. H. Getto. A physically-based particle model of woven cloth. *The Visual Computer*, 8:264–277, 1992. 10.1007/BF01897114.

[3] D. E. Breen, D. H. House, and M. J. Wozny. Predicting the drape of woven cloth using interacting particles. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, SIGGRAPH '94, pages 365–372, New York, NY, USA, 1994. ACM.

[4] R. Hooke. De potentia restitutiva. *London: John Martyn*, 1678.

[5] S. Kawabata. The standardization and analysis of hand evaluation. 1980.

[6] M. Meyer, G. Debunne, M. Desbrun, and A. H. Barr. Interactive animation of cloth-like objects in virtual reality. *The Journal of Visualization and Computer Animation*, Volume 12(Issue 1):1–12, May 2001.

[7] J. Mosegaard. Mosegaards cloth simulation coding tutorial, June 2009.

[8] S. Newton, D. Bernoulli, L. Euler, and C. Maclaurin. *Philosophiae naturalis principia mathematica*. Typis Barrillot & Filii, 1740.

[9] H. N. Ng and R. L. Grimsdale. Computer graphics techniques for modeling cloth. *IEEE Comput. Graph. Appl.*, 16:28–41, September 1996.

[10] X. Provot. Deformation constraints in a mass–spring model to describe rigid cloth behavior. In *Graphics Interface '95*, pages 147–154, May 1995.

[11] J. Weil. The synthesis of cloth objects. *SIGGRAPH Computer Graphics*, 20:49–54, August 1986.

Comparison of JavaScript libraries

Johan van der Geest
Mark Ettema

University of Groningen

Abstract— JavaScript libraries are important instruments for website developers. They simplify the development of JavaScript-based websites with code that is easier to write and maintain. With the rise of Web 2.0, that often relies on JavaScript, these libraries became increasingly popular. This research is done from the viewpoint of a web developer that has to choose the right library for a project. First we made a selection of the four most popular JS libraries, which seemed to be jQuery, MooTools, Prototype and YUI. Then we did a detailed comparison between these libraries. This was done in terms of browser compatibility, community involvement, performance, distinctive features and support for future technologies. In most tests, jQuery scored very well, especially on community involvement. However, web developers should always carefully select a library based on its features and the value it adds to a project. By reading through this paper, a web developer can easily find out which characteristics are important to him and make the decision.

Index Terms—JavaScript, JavaScript libraries, comparison, jQuery, MooTools, Prototype, YUI, Web 2.0

1 Introduction

JavaScript is an object-oriented scripting language that is used to provide client-side functionalities on websites. It allows web developers to write code that manipulates objects on-the-fly. At the moment JavaScript has a position in the top 10 of most popular programming languages [1]. The rise of Web 2.0 (with interactive user interfaces etc.) makes JavaScript a more crucial factor for web developers and browser vendors.

An advantage of JavaScript is that it is supported by the browser without the need of a plug-in. Therefore JavaScript works on almost all platforms, unlike for example Adobe Flash [2]. Browser vendors are working on better JavaScript engines for fast execution of JavaScript code [3]. This competition between browsers engines will improve the performance of JavaScript.

Traditional JavaScript is used for simple functions, such as animations. With AJAX (Asynchronous JavaScript and XML) technology [4] it is possible to interact with the server when a web page is already loaded. Nowadays JavaScript is used for more advanced functions, therefore programming in a well-structured and cross-browser compatible way becomes more important. This is one of the reasons to use a JavaScript library.

A JS library is a set of functions and utilities that makes programming in JavaScript easier. The typical features are [5]:

- *Selectors:* A selector is a short method to select a HTML element. This method can be compared with the regular `document.getElementById` function, but it also supports selecting by class name and element type.
- *DOM traversal:* With DOM traversal an element can be selected using the Document Object Model tree [6]. This means that it is possible to select a parent or child(s) from an element. For example a paragraph or a group of images.
- *DOM manipulation:* With DOM manipulation it is possible to change the content or style of a selected element.
- *Utility functions:* Most of the JS libraries have utility functions. With the use of these functions some code can be written shorter and more structured. This makes the code more maintainable.
- *Event handling:* Event handling, for example reading out a key stroke, is not implemented exactly the same in each browser. A JS library provides methods that are cross-browser compatible.
- *AJAX:* To interact with a server the data is mostly formatted as XML or JSON. With a JS library it is easier to make a request in the right format and use a callback function to process the received response.

The described features are part of almost all libraries, but what is the best one to pick as a web developer? At the moment there are only a few JS library comparisons available, but most of them compare the different features of just two libraries. The goal of this paper is to describe the differences between the four most popular JS libraries and thereby help a web developer to select the best library for his needs. Therefore we start with a selection of four JS libraries in section 2. When the selection is made we compare them in section 3. This is a detailed comparison including a performance test. Finally, in section 4, we give a conclusion that describes which library scores the best for a specific need of a web developer.

2 Selection

Before we can make a comparison between JS libraries, we need to make a selection of the most popular ones. There are dozens of JS libraries available, and not all of them are equally popular. Some of them do not offer all of the six features mentioned in the introduction (selectors, DOM traversal, DOM manipulation, utility functions, event handling and AJAX). Other projects have just started and are not as mature as the larger JS libraries. Our goal is to do an in-depth comparison of the four most popular JS libraries.

We found out that jQuery, MooTools, Prototype and YUI are the most popular JS libraries nowadays. We did this with the help of Alexa [7], a company that generates a list of the most popular websites on the internet sorted on page views. Several researches on the internet measured the popularity of JS libraries by finding out what the most popular websites in the Alexa ranking have implemented.

One of the researchers, Elie Bursztein [8], developed a crawler that went through the first 100.000 websites of the Alexa ranking. He found out that 45% of these websites use a JS library. From these websites, jQuery, MooTools, Prototype and YUI were the most commonly used JS libraries.

- *Johan van der Geest, MSc student at University of Groningen, E-mail: j.van.der.geest@student.rug.nl.*
- *Mark Ettema, MSc student at University of Groningen, E-mail: m.s.ettema@student.rug.nl.*

Figure 1 shows the difference in popularity between the JS libraries implemented by the top 100.000 websites of the Alexa ranking. The other researches [9][10] show similar results and come with the same ranking as Elie Bursztein.

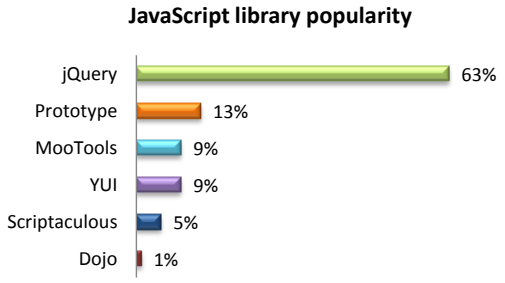


Fig. 1. JS libraries used among the Alexa top 100.000 websites.

3 Comparison

The previous paragraph justified our selection for jQuery, MooTools, Prototype and YUI as being the most popular JS libraries nowadays. We do the comparison of these JS libraries in terms of browser compatibility, community involvement, performance, distinctive features and support for future web technologies such as HTML5 and CSS3. The latest versions of the libraries are compared, which are at the time of writing: jQuery 1.5.1, MooTools 1.3.1, Prototype 1.7 and YUI 3.3.

For the first three topics (browser compatibility, community involvement and performance) we do a score-based evaluation. Based on statistical data we hand out a relative score ranging from zero to 100 for each of the topics. These 100 points are divided amongst the four libraries. We always give the corresponding equation that we used for computing the scores.

The last two topics (distinctive features and future web technologies) do not get scores as they depend more on the needs of a web developer and are harder to relate to numbers.

3.1 Browser compatibility

For web developers browser compatibility is a very important aspect. Visitors can use different web browsers and therefore a website should function the same in multiple browsers. A JS library helps with programming in a cross-browser compatible way, but the number of supported browsers differs per library.

All four libraries have support for the five most popular web browsers [11], as can be seen in Table 1. The table shows for each JS library the oldest browser version that is supported. Prototype offers the best support for older web browsers, with the exception for Opera as compared to MooTools. In contrast, jQuery mostly requires newer web browsers, which results in a lower score. Both MooTools and YUI are in between. The data was taken from the official websites of the JS libraries [12] [13] [14] [15].

Library (L)	Chrome (C)	Firefox (F)	IE (I)	Opera (O)	Safari (S)	Score (S_{bc})
jQuery	8.0	2.0	6.0	10.6	3.0	23
MooTools	4.0	2.0	6.0	9.0	3.0	25
Prototype	1.0	1.5	6.0	9.25	2.0.4	27
YUI	1.0	3.0	6.0	10.0	4.0	25

Table 1. Results of the browser compatibility by the JS libraries.

The scores are computed with equation 1. The library with the best support for older versions gets the highest score. The scores are rounded to whole numbers.

$$S_{bc}(L) = \frac{1}{3} \cdot \Big[100 - \big(\frac{C(L)}{sum(C)} + \frac{F(L)}{sum(F)} + \frac{I(L)}{sum(I)} + \frac{O(L)}{sum(O)} + \frac{S(L)}{sum(S)}\big) \cdot \frac{100}{5}\Big] \quad (1)$$

3.2 Community involvement

One of the most important factors for a JS library to be successful is the size and involvement of its community. Web developers often consult a search engine such as Google to find examples, solutions for problems or complete scripts (plug-ins) for a JS library. Some web developers are also interested in buying a book that discusses a JS library in detail.

As such, we compared the number of Google search results, available books on Amazon.com and number of questions on Stack Overflow for each of the libraries. Stack Overflow is a large Q&A (Question & Answer) site for software developers and thereby a valuable source to measure the community involvement of a JS library.

As can be seen in Table 2, the jQuery JS library is clearly a winner here. The changes are much higher to find information about jQuery than with MooTools, Prototype or YUI. The Prototype library comes on a second place, but the difference with the jQuery is very high. The third and fourth place are very close, taken by YUI and MooTools respectively.

Library (L)	Google (G)	Amazon (A)	Stack Overflow (S)	Score (S_{ci})
jQuery	29,500,000	38	24,398	72
MooTools	3,170,000	7	554	7
Prototype	10,800,000	9	1,132	14
YUI	7,490,000	4	387	8

Table 2. Results of search queries performed at March 17, 2011.

We found it important to be consistent with the search results. On Google, we used the search term "*[library name]* + JavaScript" to filter out results that are not related to the particular JS library. (For example, prototype is a common word that has different meanings [16]). On Amazon, we searched for books with the name of the library as title. We added "JavaScript" as keyword to our search query to filter out books about other subjects. Finally, on Stack Overflow, we searched on questions with "JavaScript" and the name of the library as tag.

The scores are computed with equation 2. They are rounded to whole numbers.

$$S_{ci}(L) = \big(\frac{G(L)}{sum(G)} + \frac{A(L)}{sum(A)} + \frac{S(L)}{sum(S)}\big) \cdot \frac{100}{3} \quad (2)$$

3.3 Performance

The last topic that is evaluated using scores is the performance of JS libraries. The requirement specifications of (web-based) software often include performance as one of the most important non-functional requirements. As such, the performance of a JS library is an important factor for a web developer.

We measure the performance of JS libraries in two ways. The first one is the file size of a library, which has a direct relationship with the time it takes to download the library from a web server. The earlier a library is downloaded, the earlier a website can be rendered by a web browser. The second one is the run-time performance of a library. Large websites with many user interface interactions easily require thousands lines of programming code. There could be a significant difference in run-time performance between the four JS libraries.

3.3.1 File size

Table 3 gives an overview of the JS libraries and the file sizes of their latest versions. MooTools is the smallest one and gets the most points.

Library (L)	Size (S)	Score (S_{fs})
jQuery	83.3 kB	24
MooTools	7.4 kB	32
Prototype	159.5 kB	15
YUI	36.1 kB	29

Table 3. The file size of the latest JS libraries.

On the second place is YUI. It should be noted that the file sizes of MooTools and YUI only contain the pure core of the libraries. You can customize these libraries with only the modules you need for your project. On the third place we find jQuery and the largest library is Prototype, so this one gets the lowest score.

To compare the file size we have downloaded the smallest available version of each library from the official websites. Then we calculated the scores using equation 3 and rounded it on whole numbers.

$$S_{fs}(L) = \frac{1}{3} \cdot \left[100 - \frac{S(L)}{sum(S)} \cdot 100\right] \quad (3)$$

3.3.2 Run-time performance

For a web developer it is important that his or her web application runs fast. To compare the libraries, we have done two run-time performance tests, namely Slickspeed [17] and Taskspeed [18]. Slickspeed is used to compare the speed of selecting elements, while Taskspeed compares the speed of different advanced tasks. MooTools uses a copy of Slickspeed on the official website [19] and YUI describes the tools in a presentation about performance [20]. In the two sub-sections more details about these tests and the results are described.

As can be seen in Table 4 the winner of the run-time performance tests is jQuery, followed by Prototype and YUI on a shared second place. The slowest library is MooTools and therefore it gets the lowest score.

Library (L)	**Slickspeed** (S)	**Taskspeed** (T)	**Score** (S_{rp})
jQuery	52 ms	121 ms	27
MooTools	149 ms	192 ms	21
Prototype	68 ms	134 ms	26
YUI	61 ms	111 ms	26

Table 4. Average results of the run-time performance tests.

The table shows the average time a library needs to complete a test. A lower value means a better performance and therefore gets a higher score. We calculated the scores using equation 4 and rounded it on whole numbers.

$$S_{rp}(L) = \frac{1}{3} \cdot \left[100 - \left(\frac{S(L)}{sum(S)} + \frac{T(L)}{sum(T)}\right) \cdot \frac{100}{2}\right] \quad (4)$$

3.3.2.1 Slickspeed

Slickspeed is a web application that tests the speed of JS libraries by selecting a number of DOM elements using select queries. The select queries are performed on each library and the time needed to complete each query is measured. The list of executed queries can be found in the `selectors.list` file, which is included in the Slickspeed package.

Each test is performed by us on seven browsers. We selected the most popular browser versions [21] and also included the latest version (if it was not already the most popular one). It is important that the test is performed with the browsers that the average visitors use. Figure 2 shows the popularity of the selected browsers. The results are normalized so the sum of the popularity is 100 percent.

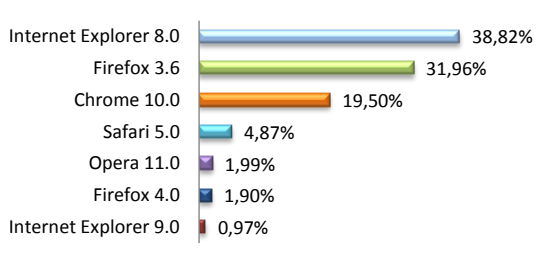


Fig. 2. Worldwide popularity of browser versions in week 11-2011.

In Figure 3 the detailed results of the test are shown. It not only indicates the performance differences between the libraries, but also shows how they perform in different browser versions. For calculating the average value, we took into account the popularity of the web browsers (see Figure 2). The results show that Internet Explorer 8.0 has the slowest Javascript engine, while Opera 11.0 has the fastest. Overall jQuery and Prototype are the fastest JS libraries for selecting DOM elements, while YUI and MooTools are the slowest.

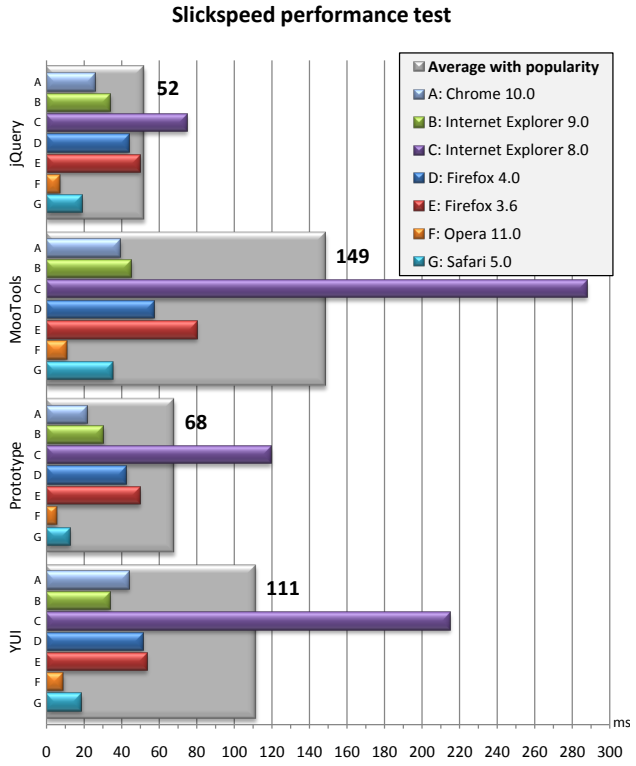


Fig. 3. Slickspeed performance test in milliseconds (lower is better).

To make the test reliable we defined the following criteria points:

- For each browser, the test is performed three times. The average value is taken from these test.
- An InPrivate browser session is used, so there is no influence by already cached data.
- All tests are performed on the same hardware and on the same operating system (Windows 7 32bit with Service Pack 1).
- Most background applications, such as virus protection are disabled.

The test itself uses JavaScript and can be started when the website is completely loaded, so there is no influence by the Internet connection speed. To avoid conflicts between the libraries each one runs in its own iFrame.

We updated Slickspeed with the latest and smallest versions of the JS libraries, as described in Section 3.3.1. The smallest versions of MooTools and YUI do not include all the functions needed to complete the tests. Therefore we downloaded the recommended version of MooTools. For YUI there is no recommended version available, so we used the YUI Configurator on the official website [15] to create a library file. We selected the following modules (other dependencies will be added automatically): `yui, node` → `node-base` and `selector-css3`. We added `Y = YUI().use('*');` at the end of the created library file. This was needed to load the selected modules.

3.3.2.2 Taskspeed

Taskspeed is based on Slickspeed, but it executes complete tasks instead of only selecting elements. The construction is the same, the time needed to complete each task is measured.

In Figure 4 the detailed results of the tests are shown. For each library there are sixteen tasks executed and the needed time is displayed in the graph. Pure DOM shows the speed without using a JS library. This is the fastest method, since there is no overhead from a JS library. Overall YUI is the fastest library, followed by jQuery and Prototype on the second and third place respectively. Again, the slowest library is MooTools.

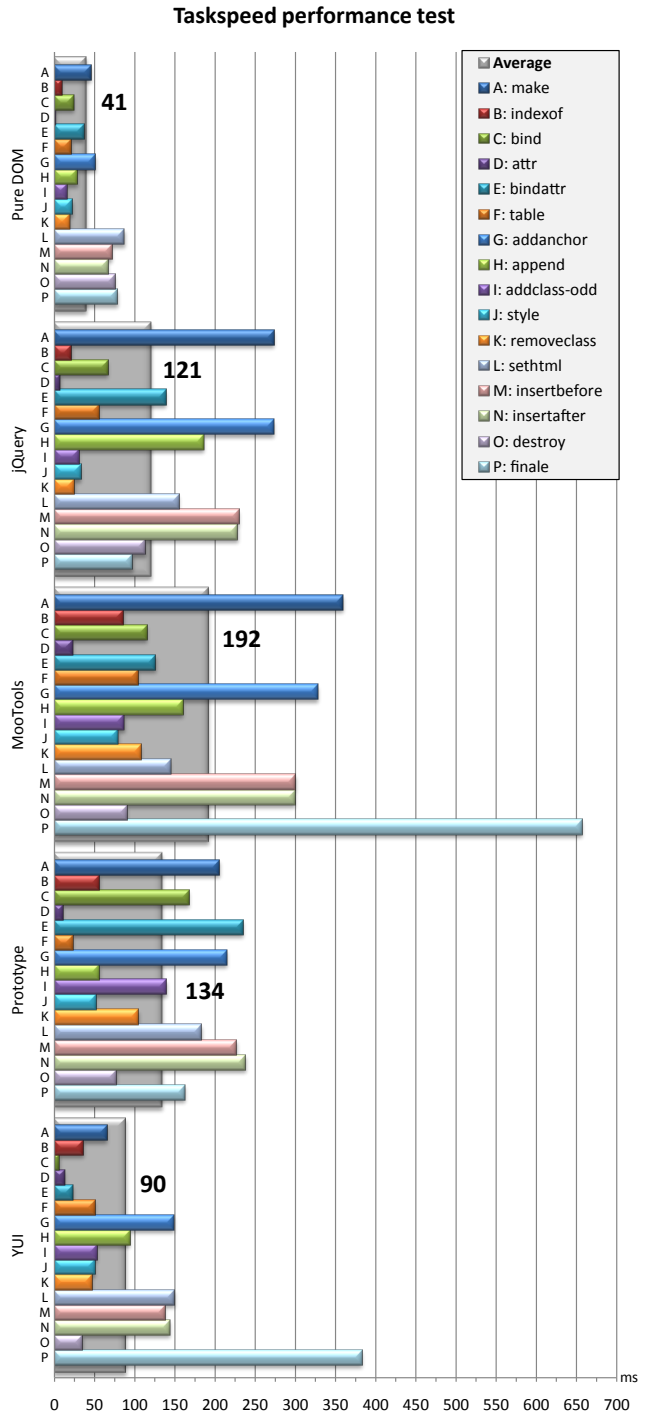


Fig. 4. Taskspeed performance test in milliseconds (lower is better).

The test is performed using the same criteria we defined in section 3.3.2.1. The different results of each browser are not shown, because this makes the graph too complex. Therefore we take into account the browser popularity by calculating the values. By showing the different tasks a web developer can compare the tasks that are important for a specific project.

A detailed description of each task can be found in the `sample-test.js` file, which is included in the Taskspeed package. In Listing 1 the description of "addclass-odd" is shown. Because the syntax of each library is different, there is test-case file for each library where the comments are replaced with the right implementation.

```
1 "addclass-odd" : function(){
2     // locate all div elements on the page
3     //  add the class "added" to those divs
4     //  add the class "odd" to the odd divs in the
          selection
5     //
6     // return the lenght of the odd found divs
7   }
```

Listing 1. Description of the test case "addclass-odd".

We have updated the JS libraries for Taskspeed to the same versions we used for the Slickspeed test. Taskspeed uses more functions than only selecting elements and therefore we have created a new version of the YUI library. This was done because not all needed functionalities were included in the version for Slickspeed. We have selected `dd` → `dd-delegate` and the other dependencies are included automatically. Because the syntax of YUI version 3 and higher differs in some cases from the previous versions, we also updated the test case file [22].

3.4 Distinctive features

Numbers cannot always give a meaning to everything. An example of such a case is the distinctive features of a JS library. For this topic we highlight the most important characteristics of each of the four JS libraries. Some libraries are more suitable to a certain type of project than another.

3.4.1 jQuery

"Write less, do more" is the slogan that jQuery uses [12]. This JS library focuses very much on simplifying HTML document traversing, event handling, animating and support for AJAX. It gives the web developer a collection of methods and allows a shorter way of writing JavaScript code in a cross-browser compatible way. Listing 2 shows a typical block of jQuery code that accomplishes a rather simple task: display a message box when the user clicks on a hyperlink.

```
1 $(document).ready(function() {
2   $('a').click(function(event) {
3     alert('You just clicked on a hyperlink!');
4   });
5 });
```

Listing 2. Displaying an alert after a hyperlink click with jQuery.

The jQuery code is short and easy to understand for a web developer. The `$(document).ready()` method assures the web developer that the code is executed after the DOM is fully loaded. The dollar sign (`$`) is an alias to `jQuery`. It is used to search through the DOM for objects and if desired, perform operations on them. The `$('a')` snippet returns all hyperlinks and `.click()` adds an event listener to these elements. A very good resource for web developers is Antonio Lupetti's cheat sheet [23], in which all of jQuery's features are summed up. Another good resource is the official documentation from jQuery, which is very complete and easy to understand.

An extension to jQuery that is worth mentioning is jQuery UI. It provides widgets, interactions, effects, and theming on top of jQuery. The extension is maintained by the developers of jQuery. You can create a custom build with the components you need on the website of jQuery UI [24].

3.4.2 MooTools

MooTools focuses on making the JavaScript API more stable and coherent. Unlike jQuery, it is less focused on "changing the way you write JavaScript". Instead, it focuses on the JavaScript programming language and seeks to streamline it, but in such a way that it not deviates from the basic principles of JavaScript. A significant portion of the core library is spent on augmenting Function, String, Array, Number, Element and other prototypes [25].

Listing 3 achieves the same thing as Listing 2, but is written using the MooTools library. The code is not as compact as jQuery, but comes much closer to traditional JavaScript code.

```
1 window.addEvent('domready', function() {
2   $$('a').addEvent('click', function(event) {
3     alert('You just clicked on a hyperlink!');
4   });
5 });
```

Listing 3. Displaying an alert after a hyperlink click with MooTools.

When your intention as a web developer is to write cross-browser compatible and streamlined JavaScript code, but not necessarily shorter code such as jQuery, then MooTools is definitely a good choice for you. Just keep in mind that the learning curve of MooTools might be a tad steep when compared to jQuery. Also, the official documentation from MooTools is not as good as jQuery's documentation.

3.4.3 Prototype

Prototype includes a lot of enhancements to JavaScript that makes writing a web application easier. For instance, it adds methods to the Function object and the Enumerable class, which provides advanced mechanisms for working with groups of objects. The DOM manipulation functions are useful, but the plugin system of Prototype is not as rich as the one from jQuery [26].

When you want to work with multiple JS libraries in one web application, then Prototype is not a very wise choice. Since Prototype changes the default behavior of JavaScript elements, it does not integrate very well, unlike jQuery, MooTools and YUI [27].

Listing 4 shows the Prototype variant of displaying an alert after the user clicks on a hyperlink. The code is not as compact as MooTools and jQuery.

```
1 document.observe('dom:loaded', function() {
2   $$('a').each(function(event) {
3     observe('click', function(event) {
4       alert('You just clicked on a hyperlink!');
5     });
6   });
7 });
```

Listing 4. Displaying an alert after a hyperlink click with Prototype.

Prototype is the default JS library in Ruby on Rails, a popular web application framework. This helped Prototype to grow in its early days. With version 3 of Ruby on Rails it is much easier to switch to another JS library. Some people think this is the cause of a slowing development process and a decrease in popularity of Prototype [28].

A popular extension to Prototype is script.aculo.us [29]. It provides a visual effect engine and some UI controls on top of Prototype.

3.4.4 YUI

The last JS library we examine is YUI, which stands for *Yahoo! User Interface*. YUI is a JS library that forces us to follow predefined coding patterns and provides good maintainability. It is designed as a framework to cover all the aspects of UI development for a complete system. When a web application contains many components with complex dependencies, then YUI is very likely a good choice [30].

Listing 5 shows the YUI variant of the previous code examples. From the four JS libraries, it is the least compact code. It can be argued that it is not as easy to understand as the other JS libraries. What would `filter:'raw'` and `.use('node')` exactly do?

```
1 YUI({filter: 'raw'}).use('node', function(Y) {
2   var showMessage = function(event) {
3     alert('You just clicked on a hyperlink!');
4   };
5 }
6 Y.on('click', showMessage, 'a');
```

Listing 5. Displaying an alert after a hyperlink click with YUI.

3.5 Future web technologies

For the future web technologies we can conclude that all the four libraries are on nearly the same level. They all have support for CSS3 and jQuery even shows on the main page that it is CSS3 compliant. Discussions and information on the official websites indicate that there is some support for HTML5. It is not always directly cross-browser compatible and sometimes there are tricks needed to let it work correctly. For each library there can be HTML5 plug-ins or code samples found, for example to control HTML5 videos.

When these new technologies become more standard and will be used by more people, we can be pretty sure that the support in the upcoming versions will be better. There is no reason to choose a specific library because of the future technologies.

4 Conclusion

JavaScript libraries have many advantages. It helps web developers to program in a cross-browser compatible way, makes code shorter and better maintainable, makes the use of AJAX technology easier and there is much more to gain. The key question that we answered in this paper is: which JS library should I pick as a web developer?

We compared the four most popular JS libraries. For browser compatibility (S_{bc}), community involvement (S_{ci}), library file size (S_{fs}) and run-time performance (S_{rp}) we did a score based comparison. The results of these comparisons are shown in Table 5, in which the average result of each library is also calculated.

Library	S_{bc}	S_{ci}	S_{fs}	S_{rp}	Average
jQuery	23	72	24	27	37
MooTools	25	7	32	21	21
Prototype	27	14	15	26	21
YUI	25	8	29	26	22

Table 5. Scores of the different comparisons and the average result.

When we purely focus on these scores we can conclude that jQuery is the best JS library to pick. It received the highest score on both community involvement and run-time performance. There is not much difference in the final scores of the other three libraries, so it really comes down to what a web developer find most importantly. It should be noted that MooTools is the only library that scores the worst in two comparison criteria, namely community involvement and run-time performance.

Other positive aspects of jQuery are the large amount of available plug-ins and the good documentation which makes it easy to learn. On the other hand, for large front-end web applications, YUI might be a better choice than jQuery. It is more suitable for large, complex architectures with many dependencies. MooTools falls in between, and is aimed at the web developer that wants to write cross-browser compatible JavaScript code, but not necessarily as short as jQuery. Prototype is perhaps best suitable for those who write Ruby on Rails web applications, although with version 3 it is easier to pick another JS library. Each of these three libraries have a steep learning curve when compared to jQuery.

Now that we have provided a very complete overview of the differences between the four most popular JS libraries, it is up to the web developer to decide which JS library will suit his or her project the best. By comparing the results and discussions in this paper, the choice for picking a JS library should now be much easier to make than one could ever do before.

REFERENCES

[1] DedaSys LLC, "Programming language popularity." http://langpop.com/. *Date accessed: Feb 2011.*

[2] Franklin Reynolds, "Web 2.0 - in your hand," *Pervasive Computing, IEEE*, vol. 8, pp. 86–88, Jan.-March 2009.

[3] Holger M. Kienle, "Its about time to take javascript (more) seriously," *Software, IEEE*, vol. 27, pp. 60–62, May-June 2010.

[4] Zhijie LIN, Jiyi WU, Qifei ZHANG, Hong ZHOU, "Research on web applications using ajax new technologies," *MMIT '08.*, pp. 139–142, Dec 2008.

[5] Joe Lennon, "Compare javascript frameworks." http://www.ibm.com/developerworks/web/library/wa-jsframeworks/. *Date accessed: Feb 2011.*

[6] Philippe Le Hgaret, Lauren Wood, Jonathan Robie, "What is the document object model?." http://www.w3.org/TR/DOM-Level-2-Core/introduction.html. *Date accessed: Feb 2011.*

[7] Alexa, "Alexa top 500 global sites." http://www.alexa.com/topsites. *Date accessed: Feb 2011.*

[8] Elie Bursztein, "45% of the popular websites use a javascript framework." http://elie.im/blog/web/45-of-the-popular-websites-use-a-javascript-framework/. *Date accessed: Feb 2011.*

[9] Royal Pingdom, "Javascript framework usage among top websites." http://royal.pingdom.com/2008/06/11/javascript-framework-usage-among-top-websites/. *Date accessed: March 2011.*

[10] Fabian Beiner, "Javascript-framework vergleich." http://fabian-beiner.de/alexa-top100-deutschland-javascript-framework-vergleich/. *Date accessed: March 2011.*

[11] StatCounter Global Stats, "Browsers stats from week 11-2011." http://gs.statcounter.com/#browser-ww-weekly-201111-201111-bar. *Date accessed: March 2011.*

[12] jQuery, "Official website." http://jquery.com/. *Date accessed: Feb 2011.*

[13] MooTools, "Official website." http://mootools.net/. *Date accessed: Feb 2011.*

[14] Prototype, "Official website." http://www.prototypejs.org/. *Date accessed: Feb 2011.*

[15] YUI, "Official website." http://developer.yahoo.com/yui/3/. *Date accessed: Feb 2011.*

[16] Wikipedia, "Prototype (disambiguation)." http://en.wikipedia.org/wiki/Prototype_(disambiguation). *Date accessed: March 2011.*

[17] Valerio Proietti, "Slickspeed project." https://github.com/kamicane/slickspeed. *Date accessed: March 2011.*

[18] Taskspeed, "Project." https://github.com/MadRabbit/taskspeed. *Date accessed: March 2011.*

[19] MooTools, "Slickspeed." http://mootools.net/slickspeed/. *Date accessed: March 2011.*

[20] Matt Sweeney, "Yui 3 performance." http://yuilibrary.com/~msweeney/yuiconf-yui3perf.pdf. *Date accessed: March 2011.*

[21] StatCounter Global Stats, "Browsers versions from week 11-2011." http://gs.statcounter.com/#browser_version-ww-weekly-201111-201111-bar. *Date accessed: March 2011.*

[22] YUI3, "Test case file." http://yuilibrary.com/~msweeney/yui-tests/taskspeed/tests/. *Date accessed: March 2011.*

[23] A. Lupetti, "Cheat sheet for jquery." http://woorkup.com/wp-content/uploads/2011/02/jQuery-1.5-Visual-Cheat-Sheet.pdf. *Date accessed: March 2011.*

[24] jQuery UI, "Official website." http://jqueryui.com/. *Date accessed: March 2011.*

[25] A. Newton, "jquery vs. mootools." http://jqueryvsmootools.com/. *Date accessed: March 2011.*

[26] seasoup, "Prototype and jquery: A code comparison." http://ajaxian.com/archives/prototype-and-jquery-a-code-comparison. *Date accessed: April 2011.*

[27] Graza, "Prototype vs jquery." http://stackoverflow.com/questions/2644556/prototype-vs-jquery. *Date accessed: April 2011.*

[28] Quora, "Why is prototype js development so slow, and is there no future for it?." http://www.quora.com/Why-is-Prototype-JS-development-so-slow-and-is-there-no-future-for-it. *Date accessed: April 2011.*

[29] script.aculo.us, "Official website." http://script.aculo.us/. *Date accessed: April 2011.*

[30] R. Akkineni, "Yui3 vs jquery." http://dsheiko.com/weblog/yui3-vs-jquery. *Date accessed: April 2011.*

Main Memory Database Systems
Opportunities and pitfalls

Wytze Hazenberg Sjoerd Hemminga

Abstract— Main memory database systems store their database in main physical memory, as opposed to traditional systems, which use disk memory. We research how main memory can be used to improve response times in database systems. While it seems obvious that performance should improve, just by using fast access physical memory, in practice, this is not always the case. As it turns out, implementation considerations can have great implications for the overall performance of a main memory system.

Index Terms—main memory database systems, database architecture, index structures, transaction handling, error recovery, two-tier.

1 Introduction

Database systems have traditionally been geared toward disk-based storage, as disk memory provided large amounts of storage at a relatively low price. In the last decades, however, chip memory sizes have grown considerably, while their price has dropped. As a consequence, using main memory for applications which traditionally used disk memory has become technically possible and economically feasible. The main incentive for using main memory instead of disk memory is the lower access times the former offers. Time-critical applications, such as telephone switchboards, require database systems with low response times. This has prompted database developers to consider Main Memory Database Systems (MMDB's), which store their data in main physical memory, for these applications.

As disk-based database systems also use main memory for caching, a very straightforward way of implementing an MMDB is increasing the cache size of such a system to fit the entire database in cache. This approach requires very little work to the database system, as it only requires changing one of its settings. It increases the system speed, as it minimizes the number of disk accesses required. However, it does have a number of problems. Every request has to pass the buffer manager to check which data is in cache and which is not, creating a lot of overhead. Moreover, the entire system stresses the efficient use of disk accesses and disk storage, rather than CPU cycles and memory. In order to use main memory as efficiently as possible, and to get as big a performance increase as possible, a more radical approach is required. Algorithms for query processing, concurrency control, and database recovery, as well as different index structures, must be reconsidered.

In section 2 we describe what index structures should be used in MMDB's and in what circumstances they offer best performance. Then, in section 3, we point out what implementation details, aimed at improving performance in disk-based systems, are not suitable for an MMDB and should be reconsidered. Details and merits of a two-tier system, which combines features of a disk-based and main memory database system, are discussed in section 4.

2 Index Structures

In order to improve the efficiency of data retrieval in tables databases use data structures called indexes. They allow the database system to quickly find data in a table, without having to scan the entire database [18]. The primary goals for a disk-oriented data index structure are to minimize the number of disk accesses and to minimize disk space usage. As a main memory index structure is contained in main memory, there are no disk accesses to minimize. Therefore, the primary goals of a main memory index are to reduce computation time and use as little memory as possible [25].

Furthermore, as relations are memory resident, a main memory index structure does not have to store actual attribute values. Instead, it may store pointers to tuples, and use them to extract values when they are needed. This has three advantages. First, every tuple pointer provides the index with access to both the attribute value and the tuple itself, reducing the size of the index as every tuple is only stored once. Second, this provides an efficient way of dealing with long fields, variable length fields, and compression techniques (which are dealt with outside the index). Third, when updates necessitate index operations, moving pointers is usually cheaper than moving attribute values.

We consider arrays, AVL trees, B trees and T trees to evaluate which structure is most suitable for use in an MMDB.

2.1 Arrays

An array is a basic data structure, in which a sequence of values is stored. While arrays use minimal space, their size is set once they are created; a new array must be created and all data copied if its size is exceeded. Therefore, their only use is in applications where the data size is known in advance or where growth is not a problem. The main drawback of using an array as an index structure is that it requires $O(N)$ data movements (N being the number of database entries) for *each* update, which renders it useless for anything but a read-only environment.

2.2 AVL trees

AVL trees are binary search trees, so every node has two children: a left child, with value smaller than or equal to itself, and a right child, with value greater than or equal to itself. AVL trees differ from ordinary binary trees in that they are balanced: for every node in an AVL tree, the height of its children can differ by at most 1 [14]. Figure 1 shows an example of an AVL tree storing integer values. It uses a binary tree search, which is very fast since it is *intrinsic* to the tree structure – no arithmetic calculations are needed.

Updates may result in an unbalanced tree, which necessitates restructuring the affected nodes by using rotation operations. For example, if a value of 54 were to be added to the tree, it would become unbalanced (figure 2). Rotations yield the tree shown in figure 3. As this restructuring may upset the balance of another node (higher up the tree), rebalancing may involve moving multiple nodes. The AVL tree's main disadvantage is its poor storage utilization. For every data item it requires one tree node, which also holds two pointers and some control information (figure 4).

2.3 B trees

B trees are a generalization of a binary search tree, in which more than two paths leave a given node [6]. A B tree node, by definition, contains between d and $2d$ keys (data items). This ensures that a node can be split into two nodes, once it has reached its maximum number of keys and another key must be added. Every internal node with k keys also contains $k+1$ pointers to child nodes; internal nodes thus contain between $d+1$ and $2d+1$ pointers. Leaf nodes only hold data items. Figure 5 shows a B tree with 2 to 4 keys per node. As one can see, the subtree to the left of a key value contains all values smaller than the key, while the subtree to the right contains values greater than the key. This feature of a B tree is used in searching: a search value S is compared to a key value K in the node. If the S is smaller than K, the

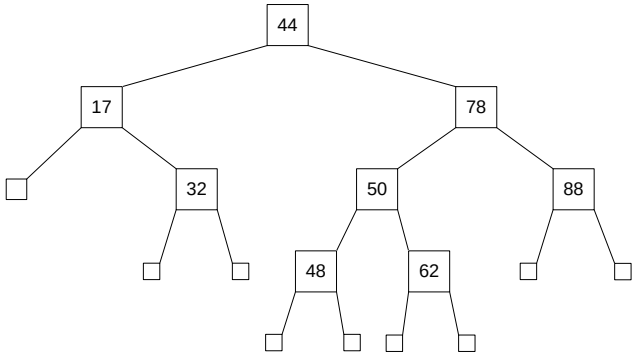


Fig. 1. An AVL tree

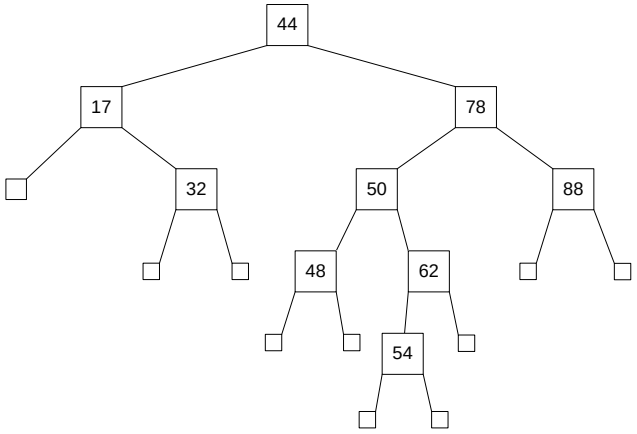


Fig. 2. Unbalanced AVL tree after insertion

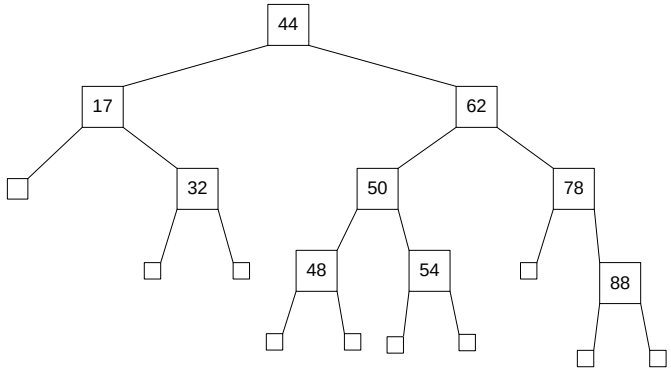


Fig. 3. Balanced AVL tree after insertion

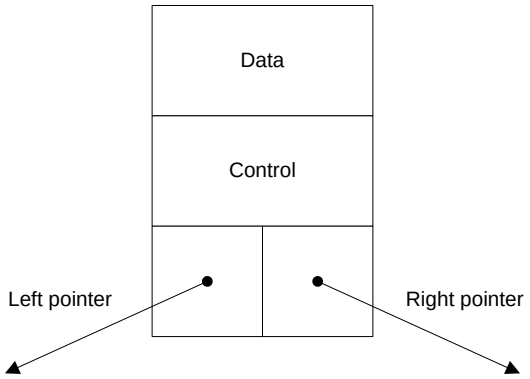


Fig. 4. An AVL tree node

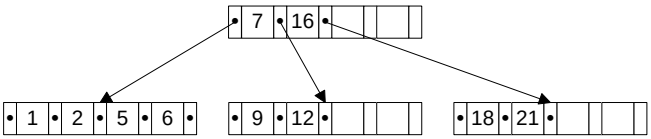


Fig. 5. A B tree

left subtree is searched; otherwise, S is compared to the next key value L. This continues until the value is discovered, or, after S is compared to the largest value in the node, continues in its right subtree. Search stops when a leaf is reached and the value is not found.

There are several variants of the B tree. Traditional database systems generally use the B^+ tree, which keeps all actual data in the leaves of the tree and uses copies of keys in the internal nodes [25]. For main memory use, however, there is no advantage to keeping all data in the leaves – it only wastes space. In a variant of the B^+ tree, the B^{link} tree, each internal node contains an additional pointer field, which links to the next node at the same level at the tree [23]. A B^* tree node contains between d and $3/2\ d$ keys, so it is at least two-thirds full[1].

B trees offer better storage utilization than AVL trees: internal nodes have better data to pointer ratios [25] (although nodes may be underfilled, wasting some space) and leaf nodes only hold data, and they comprise a large part of the tree. Using binary search, search performance is reasonably quick. Because of the variable number of keys a node can hold, nodes are more flexible in adding values as the tree does not need to be rebalanced as often as an AVL tree does.

2.4 T trees

Lehman et al. propose T trees, related to AVL trees and B trees, as a suitable data structure for use in main memory [25]. Like a B tree node, a T tree node may contain many elements, but it contains only two pointers, a left and a right child. The T tree node is displayed in figure 6. There are three types of nodes: internal nodes, leaf nodes and half-leaf nodes. Like in other trees, internal nodes have two children (both pointers are connected to a subtree), while leaf nodes do not have any children. Half leaf nodes have one child; one pointer connects to a subtree, one does not.

In a T tree, every internal node A stores a number of values in sorted order. Its first value is therefore its lower bound while its last value is its upper bound. All keys in node A have values between these bounds. Keys in the left subtree of the node all have values lower than this lower bound; the highest value in this subtree is on the right side of the subtree and is called the *greatest lower bound* of the internal node A. The right subtree contains keys with higher value; the lowest value in this subtree is on the left side of the subtree and is called the *least upper bound*. This principle is demonstrated in figure 7.

[1] In literature, a B^+ tree is sometimes referred to as a B^* tree.

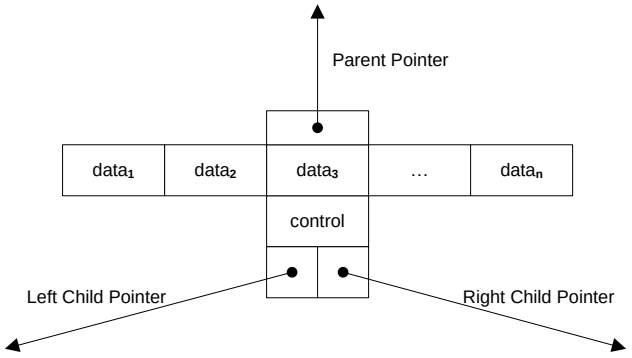


Fig. 6. T tree node

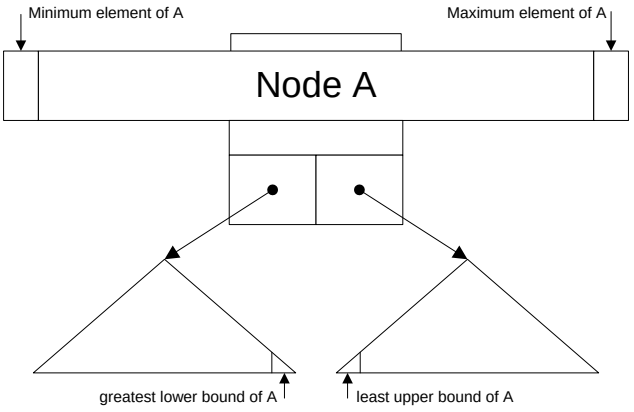


Fig. 7. T tree node with its subtrees

A search in a T tree, for every node A, looks at its boundaries. If the value it looks for is lower than the lower bound, the left subtree is searched; if it is higher than the higher bound, the right subtree is searched. Otherwise, the current node is searched. It stops if the value is found or a leaf is reached. Updating the tree resembles updating AVL trees and B trees. First, a bounding node is searched. Every node has a minimum and maximum count: the number of values stored in a node should be between those values. The value is added to or deleted from the bounding node and, only if the number of values in the node becomes unacceptable, the tree is rebalanced. Rotation operations, similar to those used in AVL trees, are used.

As a T tree is also a binary tree, it offers quick searching through binary search. Each node contains many elements, so storage utilization is good. Moreover, updates are relatively cheap, as insertion and deletion usually require moving only one node. Like in AVL trees, rebalancing may be needed, but because the nodes store more elements, and because elements can be moved inside nodes, this happens much less often.

2.5 Performance

Lehman et al. tested the data structures mentioned in sections 2.1 through 2.4 [25]. They subjected them to several tests, in which they inserted, searched for, and deleted items. Then they used range queries and sequential scans to search for multiple items, and deleted half the objects in the structure. In the tests they measured the time cost of each operation, and the storage cost of each data structure after it was built with 30,000 elements.

The results of the tests indicate that both the B tree and the T tree structure have good allround capacities. As the T tree slightly outperforms the B tree in almost all circumstances, they propose the T tree as the index structure of choice in main memory database systems.

However, in recent years, there have been studies suggesting they do not perform better than B trees and variants thereof [32]. T trees may even perform worse than a B^{link} tree, as Lu et al. show [29]. They suggest that, when performing the original test, the issue of concurrency control was not taken into consideration. Because using a T tree requires more lock operations and the overhead of locking and unlocking is high, a B^{link} tree may offer better performance in practice.

3 Database Architecture

In the following subsections we discuss some key components of database systems. The different characteristics that main memory has from disk memory have effects on almost every aspect of the database system [12].

3.1 Concurrency Control

One of the objectives in developing a database is to enable multiple users to access (search, update) data concurrently [7]. When two or more users are accessing the database simultaneously, and at least one is updating data, a mechanism is needed to prevent interference that can result in inconsistencies. Traditional database systems use transactions to do so: when one user updates, the data involved is locked and another cannot access the data until the transaction (i.e. the update) is finished. This method affects performance, as setting and releasing locks costs CPU time. Furthermore, lock contention occurs when one process tries to access data that is locked by another process, and transactions are queued.

In order to reduce contention, system design usually focuses on locking only small granules of data. Since main memory access is much faster than disk access, however, transactions complete more quickly in main memory databases, making lock contention less of an issue. Garcia et al. suggest that in a main memory system, locking granules can be much larger, maybe as large as the entire database [11, 28]. The resulting system uses serial transaction execution, which reduces overhead in setting and releasing locks and coping with deadlock. Serial execution does have a number of drawbacks, however. First, lock contention is still a problem when executing long transactions. Also, in multiprocessor systems, smaller lock granularity may be needed to improve performance.

The locking mechanism can also be optimized in main memory systems. In traditional systems, data objects are on disk and do not contain lock information. Instead, all locking information is stored in a memory resident hash table. If the data objects are in memory themselves, some locking information may be attached to them. Garcia et al., among other things, suggest adding a lock bit to every object. If a transaction wants to modify the object, it checks the lock bit to see if the object is locked. If it is not, it sets the lock bit and executes. Afterwards, it releases the lock bit. Other transactions that may be waiting for the lock to be released would still have to be administrated, but in most situations, the number of instructions needed for locking would be greatly reduced.

3.2 Commit Processing

One of the problems of main memory is its volatility. Data that is stored on disk is resident; it remains on disk unless some explicit action (deleting, overwriting) removes it. A power failure, for example, does not affect data that has been written to a disk, whereas data that is stored in main memory is lost in that case. Also, data can be stored in an array of disks. If one disk fails, it can be fixed without affecting the content on other disks, so only a fraction of the database (or, when a redundant array technology such as RAID is used, no data at all) needs to be restored from backup, while the rest of the database is still accessible. When a memory board fails, it usually means the entire system fails and the entire database is lost. It is therefore essential to have a backup copy of the database, and to keep a log of transactions

executed. In this log, the activity of every transaction is written before it commits [16].

As the log needs to be placed in stable storage, this may however affect the performance of the database system. If every transaction has to wait for a stable write to a hard disk before it can commit, response time drops and, if the log becomes a bottleneck, throughput is also reduced. These problems also exist in disk-based database systems; however, their impact is greater as in an MMDB, logging is the only disk operation in every transaction.

There are several ways of dealing with this problem. First, a portion of the log may be held in a small amount of non-volatile memory. The system first writes transaction details into this memory, so the transaction can commit. Then, data from the stable memory is copied to disks by a separate process, not interfering with database access operations. This greatly improves response time; however, it still leaves the system vulnerable to a log bottleneck. To relieve the threat of bottlenecks, group commits can be used: records of several transactions accumulate in main memory, until all are flushed to the log disk in a single disk operation.

3.3 Data Representation

Main memory systems can take advantage of pointers to data values [31, 37] for data representation. The use of pointers is space efficient when large values appear multiple times in a database. The data is only stored once and is referenced by memory pointers. Pointers can also simplify the handling of variable length data since variable length data can be represented using pointers into a heap [26, 35].

3.4 Query Processing

Sequential access is not significantly faster than random access in a main memory system, query processing techniques that take advantage of faster sequential access lose the advantage [1, 24, 31]. When relational tuples are implemented as a set of pointers to the data values, some relational operations can be performed very efficiently [31]. This because the data exists in main memory, and since they are compact data structures it can speed up queries. Query processors for main memory systems must focus on processing costs instead of minimizing disk access [37]. A difficulty is that processing costs can be difficult to measure in a complex data management system. Costly operations, like creating an index or copying data, must first be identified, and then strategies must be designed to reduce their occurrence. Operation costs may vary from system to system, so that an optimization technique that works well in one system may perform poorly in another.

3.5 Recovery

Backups of main memory systems must be maintained on disk or other stable storage to insure against loss of the volatile data. Recovery has several components, the first being the procedure used during normal database operation to keep the backup up-to-date, and the second being the procedure used to recover from a failure. Most systems that use a log for commit processing also perform backups or checkpoints to limit the amount of log data that must be processed to recover from a failure [26, 8, 19, 34], this is called check pointing [12]. In a main memory system, check pointing and failure recovery are the only reasons to access the disk copy of the database. Disk input/output should be performed using a very large block size. Large blocks are more efficiently written [12].

After a failure, a main memory system must restore its data from the log which is stored in a disk backup. If the database is large, simply transferring the data may take a long time. One possible solution to this problem is to load blocks of the database on users request until all of the data has been loaded [26, 17]. It is not clear how much of an improvement this will provide in a high-performance system which must handle thousands of transactions in the seconds after the database has recovered.

Another possible solution to the database restoration problem is to use disk striping or disk arrays [10, 33]. The database is then spread across multiple disks, and it is read in parallel. One problem is that, using multiple disks, a single bus or path from the disks to main memory becomes the bottleneck, increasing response times. A possible solution to this problem is to create independent paths from the disks to main memory.

4 Two-tier systems

Many researchers have studied several techniques such as indexing [21, 20], buffer management [4, 2], materialized view techniques [13, 30], etc. in order to provide low response times. These traditional disk-oriented techniques could not provide low response times like main memory systems can. Because of the decreasing price of main memory, some researchers argued that certain application databases will soon fit entirely in a main memory database [22, 5]. Storing a database in main memory require different techniques to optimize response times. Several memory-oriented techniques have been studied such as record format, page structures, indexing techniques, etc [25, 38]. These techniques perform significantly better than traditional systems when all data is stored in the memory buffer. However, main memory systems have restrictions on database size. User data has grown at an even faster rate than memory size can increase. It seems unlikely that the whole database of large applications will ever fit entirely into a main memory database. The present day requirements of a database system are providing low response times and handling large amounts of data. Main memory and traditional systems cannot fulfill both these requirements. Of course, an attempt can be made by using the two systems for one application. This is discussed in section 4.1.

One approach of handling these requirements is using a multi-level system [27, 36]. These systems store data in several different levels. For low response times, some parts of the data are stored in main memory. When handling large amounts of data, some parts of the data are stored on a hard disk, tape or even on an off-site network device, like traditional systems would store the data. Storing an entire database in main memory can be inefficient, because certain data is required to be accessed more often than other data would. The solution is to store frequently accessed data in main memory and less frequent data on disk. The different system components are discussed in section 4.2. This approach is called a two-tier system, as proposed by Sang-Hun Eo et al [9]. Two-tier as in two levels of storage media. Several tests were processed and the results are shown in section 4.3.

4.1 Architectural design issues

Many already constructed databases are complex and the amount of stored data can be huge. It is not feasible to reconstruct the entire database, and use the memory buffer, to provide low response times. This operation would require much time and effort. Some data in a database requires low response times, while for other data this is less important [9]. The normal way was to use two database systems. One system to handle all the data and one system to provide low response times, respectively the traditional system and the main memory system. In these environments, the traditional database stored all the data and some parts are duplicated to be handled by the main memory system. However, this concept has some problems. Problems like preparation, synchronization, efficiency and the existence of applications which use one system [9]. Application programmers needed to identify which system to reference for frequent and non-frequent required data. They needed to modify their existing applications in order to benefit from the low response times. These two systems require a synchronization module to handle data manipulations. Both systems store duplicate data, which means they do not efficiently use available resources.

4.2 Two-tier components

As proposed by Sang-Hun Eo et al., the two-tier system can provide low response times and handle massive amounts of data [9]. The proposed system consists of three major components: Disk Storage Manager, Memory Storage Manager and the United Query Processor. Basically, the whole database exists on disk. Some parts of data are duplicated, using snapshots, in main memory for low response times

using memory oriented techniques. The united query processor handles the data requests and locates the data in memory or on disk. There are three types of queries: memory query, disk-query and the hybrid query. The memory and disk query is handled by the memory storage manager and the disk storage manager, respectively. The hybrid query uses a combination of partial memory data and partial disk data. When the two-tier system detects that data is frequently requested, it stores snapshots for fast access in the future. The system can also detect periodic requests, then snapshots are updated periodically. The least recently used strategy is used to manage the memory buffer efficiently.

4.3 Testing

Sang-Hun Eo et al. performed a system test with two datasets, 10,000 and 1,000,000 rows, and four queries [9]. Only the second query was using the available index structures. The first and second query selected 1% of the first dataset. A view was used to select the first 10,000 rows of the second dataset. Queries three and four requested data from that view. Query three requested all of the view's data and query four only one specific row.

In all experiments, the disk storage manager was used first. The response times of the disk storage manager were almost identical in comparison to a traditional system. Main memory snapshots were created after 300 requests for general queries and 6 requests for view queries. The first query had an average execution time of ~275ms per request. After the creation of a snapshot the average execution time was decreased to ~100ms per request. The second query, with indexing, had an average execution time of ~130ms and after 300 requests an average execution time of ~60ms. Query 3 and 4 took a long time to execute, scanning the 1,000,000 rows, before the snapshot was created. After the creation of the snapshot the system only scanned 10,000 rows from the dataset for query 3. Query 4 only needed to find one row, so execution times were even shorter. The average execution time of query 3 decreased from ~7300ms to ~5500ms per request, after creation of a snapshot. Query 4 decreased from ~240ms to less than ~25ms per request.

5 Discussion

Although the first suggestions of a main memory database system are from the 1980's, only in recent years, more commercial implementations of true MMDB's became available. We think there are several reasons for this. First, as we showed in sections 2 and 3, the entire database system needs to be reconsidered, redesigned and rebuilt in order to benefit most from fast and volatile main memory. This may make database developers and their customers hesitant to adopt this technology. Second, hardware has evolved considerably. Most importantly, CPU speeds have grown at a much faster rate (60% per year) than main memory speeds (10% per year) [32]. This means that optimizing a database system for cache memory instead of main memory may, and in some circumstances does, give better results, as Rao and Ross showed [32]. Third, there are several new developments for optimizing database systems that show promising results, such as NoSQL, a database system for efficient handling of data over distributed nodes [3]. We think further research needs to be done to compare the performance of main memory and cache conscious database systems, and other approaches, in both distributed and non-distributed applications.

In recent years, mobile devices like cell phones and music players became more commonplace and more sophisticated. While becoming more sophisticated, these devices have also started processing and storing much more data, requiring some data structure and eventually a database to do so. These devices usually do not have a hard disk, but instead rely on both main memory (very fast, but small and volatile) and flash memory (reasonably fast and large, and non-volatile). We think that database systems for such devices, which are neither truly main memory, nor disk-based database systems, may borrow a lot from MMDB research.

6 Conclusion

In database systems, low response times are important, if not crucial, for the overall quality of the system. With its fast and nonlinear access characteristics, a database in main memory may deliver this quality. For best results, however, it is not enough to merely use a conventional, disk-based database system and use it in main memory. Several factors, such as the index structure used, the way transactions are handled and how the system recovers from errors need to be reconsidered. In practice, because disk access is still necessary for daily operation of a database system, a true main memory database system is impractical for common use.

Gray et al. proposed a rule [15], which describes which data generally should be memory resident. By analyzing the price of accessing data in memory versus accessing data on disk, they argued that data that are referenced every 5 minutes or more should be memory resident. Garcia et al. noted [12] that, as the price of a byte of main memory drops relative to the cost of disk accesses per second, this boundary increases. They expected that it would continue to increase in the future.

As it is impractical to implement a true main memory database system but increasingly economically feasible to use main memory in database systems, we have seen the advent of two-tier systems. Such a system classifies different types of data based on how frequently they are referenced. New developments focused on optimization for CPU cache or taking into consideration the characteristics of flash memory, may also profit from this approach. In these models, data that is often referenced is optimized for quick access, while other data is optimized for disk storage. A database system for mobile devices may, for example, optimize some data for main memory and other data for flash memory, while a desktop system considers CPU cache, main memory and disk memory for different types of data. As a multi-tier system automatically determines which data require low response times and which do not, existing database structures can be used. Therefore, a multi-tier system can provide low response times in existing applications without requiring major changes in system architecture.

References

[1] D. Bitton, M.B. Hanrahan, and C. Turbyfill. Performance of complex queries in main memory database systems. In *proceedings Int. Conf. on Data Engineering*, pages 72–81, February 1987.

[2] Stephane Bressan, Chong Leng Goh, Beng Chin Ooi, and Kian-Lee Tan. A framework for modeling buffer replacement strategies. In *Proceedings of the ninth international conference on information and knowledge management*, November 2000.

[3] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems*, 26, June 2008.

[4] Zhifeng Chen, Yan Zhang, Yuanyuan Zhou, Heidi Scott, and Berni Schiefer. Empirical evalutation of multilevel buffer cache collaboration for storage systems. In *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, volume 33, June 2005.

[5] Kong-Rim Choi and Kyung-Chang Kim. T*-tree: a main memory database index structure for real time applications. In *Proceedings of the Third International Workshop on Real-Time computing systems application (RTCSA '96)*, 1995.

[6] Douglas Comer. The ubiquitous b-tree. *Computing Surveys*, 11(2):121–137, June 1979.

[7] Thomas Connolly and Carolyn Begg. *Database Systems: A Practical Approach to Design, Implementation, and Management*. Pearson Education, fourth edition, 2005.

[8] M.H. Eich. A classification and comparison of main memory database recovery techniques. In *proceedings Int. Con. on Data Engineering*, pages 332–339, February 1987.

[9] Sang-Hun Eo, Yan Li, Ho-Seok Kim, and Hae-Young Bae. Two-Tier Storage DBMS for High-Performance Query Processing. *Journal of Information Processing Systems*, 4(1):9–15, March 2008.

[10] D. Patterson et al. RAID: Redundant arrays of inexpensive disks. In *proceedings ACM SIGMOD conf.*, June 1988.

[11] H. Garcia-Molina and K. Salem. High performance transaction processing with memory resident data. In *proc. int. Workshop on High Performance Transaction Systems*, December 1987.

[12] Hector Garcia-Molina and Kenneth Salem. Main memory database systems: An overview. *IEEE Transactions on Knowledge and Data Engineering*, 4(6):509–516, December 1992.

[13] Jonathan Goldstein and Per-Ake Larson. Optimizing queries using materialized views: a practical, scalable solution. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, November 2000.

[14] Michael T. Goodrich and Roberto Tamassia. *Algorithm Design*. John Wiley & Sons, first edition, 2002.

[15] J. Gray and F. Putzolu. The 5 minute rule for trading memory for disc accesses and the 10 byte rule for trading memory for cpu time. In *Proceedings 1987 ACM SIGMOD Conference*, May 1987.

[16] J. Gray and A. Reuter. Transaction processing: Concepts and techniques. *Morgan Kaufmann*, 1992.

[17] L. Gruenwald and M.H. Eich. MMDB reload algorithms. In *proceedings ACM SIGMOD conf. Denver*, pages 397–405, May 1991.

[18] Wenming Guo and Zhiqiang Hu. Memory database index optimization. In *2010 International Conference on Computational Intelligence and Software Engineering (CiSE)*, December 2010.

[19] R.B. Hagmann. A crash recovery scheme for a memory-resident database system. *IEEE Trans. Comput.*, C-35:839–842, September 1986.

[20] Sven Helmer and Guido Moerkotte. A performance study of four index structures for set-valued attributes of low cardinality. *International journal on Very Large Data bases*, 12(3), October 2003.

[21] Bijit Hore, Hakan Hacigumus, Bala Iyer, and Sharad Mehrotrasss. Indexing text data under space constraints. In *Proceedings of the thirteenth ACM international management CIKM '04*, November 2004.

[22] Minwen Ji. Affinity-based management of main memory database clusters. *ACM Transactions on internet technology (TOIT)*, 2(4), November 2002.

[23] Philip L. Lehman and S. Bing Yao. Efficient locking for concurrent operations on b-trees. *ACM Transactions on Database Systems*, 6(4):650–670, December 1981.

[24] T.J. Lehman and M.J. Carey. Query processing in main memory database systems. In *proceedings ACM SIGMOD conf.*, May 1986.

[25] Tobin J. Lehman and Michael J. Carey. A study of index structures for main memory database management systems. In *Proceedings of the Twelfth International Conference on Very Large Data Bases*, pages 294–302, August 1986.

[26] Tobin J. Lehman and Michael J. Carey. A recovery algorithm for a high-performance memory-resident database system. In *proceedings ACM SIGMOD conf., San Francisco*, volume 16, pages 104–117, May 1987.

[27] Tobin J. Lehman, J. Shekita, and Luis-Felipe Cabrera. An evoluation of starburst's memory resident storage component. *IEEE Transactions on knowledge and data Engineering*, 4(6), December 1992.

[28] K. Li and J.F. Naughton. Multiprocessor main memory transaction processing. In *proceedings Int. Symp. on Databases in Parallel and Distributed Systems*, pages 177–189, February 1987.

[29] Hongjun Lu, Yuet Yeung Ng, and Zengping Tian. T-tree or b-tree: main memory database index structure revisited. In *Proceedings 11th Australasian Database Conference*, January 2000.

[30] James J. Lu, Guido Moerkotte, Joachim Schue, and V.S. Subrahmanian. Efficient maintenance of materialized mediated views. In *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, 1995.

[31] P. Pucheral, J.-M. Thevenin, and P. Valduriez. Efficient main memory data management using the dbgraph storage model. In *Proceedings of the 16th conference on Very Large Data Bases*, pages 683–695, 1990.

[32] Jun Rao and Kenneth A. Ross. Cache conscious indexing for decision-support in main memory. In *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases*, September 1999.

[33] K. Salem and H. Garcia-Molina. Disk striping. In *proceedings Int. Conf. on Data Engineering*, pages 336–342, February 1986.

[34] K. Salem and H. Garcia-Molina. Checkpointing memory-resident databases. In *In proceedings Int. Conf. on Data Engineering*, pages 452–462, February 1989.

[35] K. Salem and H. Garcia-Molina. System m: A transaction processing testbed for memory risident data. *IEEE Trans. Knowl. Data Eng.*, 2(1):161–172, March 1990.

[36] Michael Stonebraker. Managing persistent objects in a multi-level store. *SIGMOD Conference*, pages 2–11, 1991.

[37] K.-Y Whang and R. Krishnamurthy. Query optimization in a memory resident domain relational calculus system. *ACM Trans. Database Systems*, 15(1):67–95, March 1990.

[38] Ying Xia, Sung-Hee Kim, Sook-Kyoung Cho, Kee-Wook Rim, and Hae-Young Bae. Dynamic versioning concurrency control for index-based data access in main memory systems. In *Proceedings of the tenth international conference on Information and knowledge management*, October 2001.

Comparison between NoSQL Distributed Storage Systems

Elmer Jansema, Jan Thijs

Abstract—The field of databases is dominated by relational databases which use a Structured Query Language (SQL). However these databases fail to provide native scalability and replication capabilities. Therefore so called NoSQL distributed storage systems (DSS) were introduced. In this paper we explore three of such systems; Dynamo [6] created by Amazon, Bigtable [4] created by Google and Cassandra [14] created by Facebook. We compare these systems with respect to their system architecture, data model and provide insight into their practical uses. The most important findings of our research are the aspects of well-designed NoSQL DSS; use multiple nodes for easy replication and fault tolerance, a sufficiently powerful data model, and be open source.

Index Terms—NoSQL, Dynamo, Bigtable, Cassandra, Distributed storage systems, Replication.

1 Introduction

The field of databases is dominated by relational databases which use a Structured Query Language (SQL) for storing and retrieving data. These databases are an appropriate solution for many applications which require data storage, however they lack the ability to scale natively. If a user wants to scale, he has to resort to other systems which add caching and/or partitioning features. Another problem SQL databases have is efficiently handling large data sets, in particular the aggregation of data from multiple large tables can become a slow process.

To solve these problems research has been done to design systems which can deal with these issues. A commonality between these systems is the lack of SQL, the ability to be highly available, and their aim to easily deal with partitioning of data. Because of these commonalities these systems are called NoSQL distributed storage systems (DSS).

Amazon and Google have both introduced their own proprietary NoSQL DSS called Dynamo [6] and Bigtable [4]. These systems are not for public use but are used internally by the respective companies to power some of their internal and external services. Facebook introduced the Cassandra project [14], which was made open source and is available for public use.

In this paper we present a comparison between the three previously mentioned systems. We focus on the system architecture and the data model each system uses. Furthermore we present situations in which the systems succeed and in which situations they fail.

In section 2 the system architecture of each system, and how they can achieve high availability and be distributed, is discussed. Section 3 discusses the respective data models of each system. The practical use is discussed in section 4 and in section 5 and 6 a discussion and conclusion are provided.

2 System architecture

NoSQL distributed storage systems (DSS) were designed to achieve high availability for systems which have a high user load most of the time. Structured Query Language (SQL) databases do not offer easy means to cope with this issue, therefore Amazon, Google and Facebook have developed their own storage systems. These systems are called Dynamo (Amazon), Bigtable (Google) and Cassandra (Facebook). Facebook, for example, wanted to offer their users the ability to search their inbox. Because Facebook has a large globally spread userbase, their servers are distributed across the globe to offer their users fast access to their content. This requires a storage system which can be partition tolerant and should offer high availability. In this section we describe how Dynamo, Bigtable and Cassandra solved the issues introduced by these high demands.

One of the first design decisions which has to be made is how to structure the servers, which host the NoSQL DSS, within a network. Both Dynamo and Cassandra use nodes which are structured in a ring network [6, 14]. A node is a logical unit in the architecture which is responsible for its set of data, a physical machine can have multiple nodes. In a ring network each member is connected to exactly two other nodes, forming a ring (see Figure 1). A group of nodes that work together to serve data is called a cluster.

When data gets distributed across multiple servers the chances of losing data increases due to increased risk of server failure, thus a NoSQL DSS should be fault tolerant. One of the key factors for dealing with fault tolerance is replication. If data is replicated on several different nodes, failure of one node does not imply lost data. Replication in Dynamo and Cassandra works as follows; when a data item is inserted into either of these systems, the data item is assigned a key. Each key k is assigned to a node which is in charge of replicating the data items that fall within the key range (see section 3) it is responsible for, such a node is called a *coordinator node*. Besides storing the data item locally, the coordinator node also replicates these keys at the $N-1$ clockwise successor nodes in the ring network. This implies that each node has replicated data that falls in the range between itself and its N^{th} predecessor. Since nodes can be on the same physical machine, Dynamo and Cassandra also provide means to make sure the nodes are spread across multiple machines, thus mitigating machine failure. When the node responsible for k fails, the systems fall back to a *preference list* to communicate with the next responsible node. A preference list is a list which stores the nodes that are responsible for a certain key. Each node in the network has the ability to create a preference list.

Dynamo adds flexibility to fault tolerance by offering two parameters to tweak the replication system. These parameters are R and W; R stands for the minimum number of nodes that must participate in a successful read operation and W stands for the minimum number of nodes that must participate in a successful write operation. When R is set to a low value, the performance of read operations increases however this has a negative effect on consistency. When R is set to a high value this has the inverse effect. Setting W to a high value guarantees replication, setting it to a low value creates an *always writable* system.

Cassandra extends its ability to handle faults with replication policies. There are three different policies; Rack Unaware, Rack Aware and Datacenter Aware. Depending on the selected policy, nodes are chosen to replicate data to. In the case of the Rack Unaware strategy, the coordinator node chooses the $N-1$ successive nodes on the ring and ignores if they are on physically separated machines. When a Rack or Datacenter Aware policy is chosen the Cassandra system elects a leader among its nodes using a system called ZooKeeper [11].

- *ing. E. Jansema is MSc. Computing Science student (Software Engineering and Distributed Systems) at the University of Groningen, e-mail: e.jansema@student.rug.nl*
- *ing. J. Thijs is MSc. Computing Science student (Computing Science in Business & Policy) at the University of Groningen, e-mail: jthijs@gmail.com*

When a node joins the cluster, it contacts the leader who tells the node for which key range it is the replica. The leader makes sure that no node is responsible for more than $N-1$ key ranges in the ring. The metadata, containing the key ranges a node is responsible for, is cached locally at each node and inside ZooKeeper. This way a node can crash and recover, and it would still know what key ranges it was responsible for. In the case of a Datacenter Aware strategy the leader also makes sure that every row in the data model (see section 3) is replicated across multiple datacenters thus mitigating datacenter outage.

Besides replicating data, Cassandra and Dynamo also provide means of detecting failed nodes. Failure detection is a mechanism by which a node can locally determine if any other node in the system is available and is also used to avoid attempts to communicate with unavailable nodes.

Cassandra uses a modified version of the Φ accrual failure detector [14]. The accrual failure detector emits a value, Φ, which represents a suspicion level for a monitored node. Φ is used as a threshold to decide when a node is suspected of being unavailable. When Φ is 1, then the likelihood of a mistake (suspecting a node being unavailable while it is available) is about 10. The likelihood decreases as Φ increases; 1 with $\Phi = 2$, 0.1 with $\Phi = 3$, and so on. Φ is based on the distribution of arrival times of gossip messages [7] between other nodes in the cluster. Gossip messages are messages which get sent periodically by nodes to neighbouring nodes. Every node in the system stores the arrival times and uses a sliding window on this dataset to calculate Φ. The value of Φ is dynamically adjusted to reject network and load conditions at the monitored nodes.

Dynamo uses a local notion of failure detection [6]; node *A* may consider node *B* failed if *B* does not respond to *A*'s messages (even if *B* is responsive to node *C*'s messages). Through general communication, *A* discovers that *B* is unresponsive when *B* fails to respond to a message. To be able to continue processing requests, *A* redirects the requests meant for *B* to alternative nodes. *A* will periodically check if *B* becomes responsive again. This is a form of a decentralised failure detection protocol which, like the Φ accrual failure detector of Cassandra, uses gossip messages to enable each node in the system to learn about the arrival (or departure) of other nodes [10].

Bigtable uses a completely different approach compared to Cassandra and Dynamo with respect to network structuring as well as how it handles faults. It has three major components [4]; a library that is used by every client of Bigtable, one master server, and many tablet servers. A tablet is a range of rows used as a unit of distribution and load balancing. Tablet servers contain different tablets and these servers can be added or removed dynamically from a cluster to accommodate changes in workloads. The tablet server handles read and write requests to its own tablets and also splits tablets that have grown too big. The master server is responsible for assigning tablets to tablet servers, detecting the addition and expiration of tablet servers and balancing the load of tablet servers. In addition, it handles schema changes such as table and column family creations (see section 3). Data does not move through the master server; clients communicate directly with tablet servers for reads and writes. Because Bigtable clients do not rely on the master server for tablet location information, most clients never communicate with the master server.

Although the goals of Bigtable are not providing a decentralised control and fault tolerant system, it does provide some means to prevent faults. Bigtable uses a highly-available and persistent distributed lock service called Chubby [2]. "A Chubby service consists of five active replicas, one of which is elected to be the master and actively serve requests. The service is live when a majority of the replicas are running and can communicate with each other. Chubby uses the Paxos algorithm [3, 15] to keep its replicas consistent in case of a failure and provides a namespace that consists of directories and small files. Each directory or file can be used as a lock, and reads and writes to a file are atomic. The Chubby client library provides consistent caching of Chubby files and each Chubby client maintains a session with a Chubby service. A session expires if the client is unable to renew the session lease within the lease expiration time. When a session expires, it loses any locks and open handles. Chubby clients can also register callbacks on Chubby files and directories for notifications on changes or session timeouts. If Chubby becomes unavailable for an extended period of time, Bigtable becomes unavailable." [1]

In this section we described the system architecture of Dynamo, Bigtable and Cassandra. The next paragraph will give a brief summary.

- *Network structure:* Dynamo and Cassandra use a ring network, in such a network there is no distinct leader. This can be seen in the way Dynamo and Cassandra work, for each data item inserted there is a coordinator node which is responsible for replication of that data item.

  Bigtable uses a client/master structure, which means there is one master server controlling multiple client servers. Because of the number of responsibilities the master server has, a failure of the master server would cause the system to become insufficient over time.

- *Replication:* Through the use of coordinator nodes, Dynamo and Cassandra both replicate data across multiple nodes. Also a preference list is used to store all nodes responsible for a certain data item, this allows the system to handle node failures. Dynamo offers two parameters, R and W, to control the replication system. Cassandra uses replication policies which allows the user to control the replication system.

  Bigtable does not provide a native replication system, if replication is needed clients are forced to implement their own.

- *Failure detection:* Dynamo uses a gossip protocol to learn about nodes that are unresponsive and entering or leaving the network. Cassandra uses a modified version of the Φ accrual failure detector, which calculates the suspicion level for each monitored node based on gossip messages. This allows the system to estimate if a node is available. Bigtable detects failures through the master server.

Given these system architectures we can say Bigtable clearly has a single point of failure; its master server. The master server has too many key responsibilities and failing to provide its services will significantly influence the performance of the system. This however was a conscientious design decision. Dynamo and Cassandra, on the other hand, offer a decentralised system without a single point of failure. When one node fails, there is usually another node available to provide the data. Since Dynamo and Cassandra are very similar in their system architecture, we can not say one is better than the other.

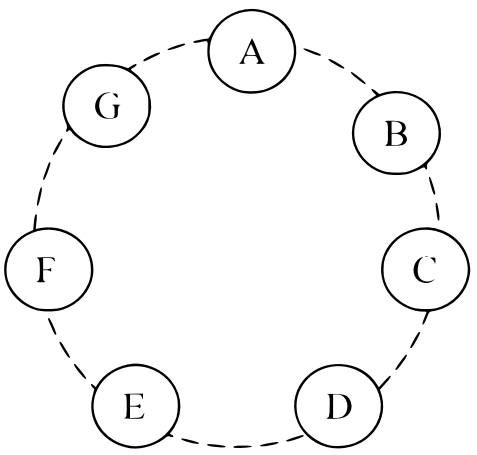


Figure 1. A visualisation of a ring network [6].

[1] From *Bigtable: A Distributed Storage System for Structured Data* [4]

3 DATA MODELS

In this section we will discuss the problems in current Structured Query Language (SQL) databases and the solutions Dynamo, Bigtable and Cassandra offer to solve these problems. We will highlight how data is stored and can be retrieved, and which data structures are used to store the data.

To understand the problem with SQL databases, we first have to look into how they work. A SQL database stores its data in multiple tables. When data needs to be retrieved from the database, a user executes a query. Because tables are usually related to each other, often the cartesian product [23] needs to be calculated. Calculating the cartesian product is useful where data from two different tables is related and needs to be merged. However calculating the cartesian product can become slow when large datasets are used, since every row from table *A* needs to be merged with every row from table *B*. The minimum result is an exponential growth of the number of data items and this exponent will grow with every table that is added to the query. To solve this problem, NoSQL distributed storage systems (DSS) use simplified data models.

Dynamo uses a data model using only keys and values, which is why it is called a highly available key-value store [6]. For storing and retrieving data it offers simple read and write operations. Each value is stored as a binary object with a unique key and has no relations with other values. The lack of relationships makes it possible for the values to be retrieved efficiently. For storing the values and keys, Dynamo has a pluggable persistence component which offers the choice to pick the database best suited for an application's access patterns. The databases that are in use are Berkeley Database (BDB) Transactional Data Store [18], BDB Java Edition [18], MySQL [17], and an in-memory buffer with a persistent backing store. Because of the variable storage engine Dynamo does not have just one binary object size limitation, since the size limitation is depended on the storage engine being used.

Bigtable is a sparse, distributed, persistent multi-dimensional sorted map [4] indexed by a row key, column name, and a timestamp. Each row key has a column family which contains a set of columns, as illustrated in Figure 2. The cells allow multiple versions of the data to be stored which are indexed by timestamps. Data from a cell can be retrieved by travelling down the hierarchy. The column names use the syntax *family:optional_qualifier*. Alternatively timestamps can be used to retrieve older data from the cells. The row keys and column names can be arbitrary strings. The row keys are maintained in lexicographic order by Bigtable. Assigning timestamps to the data can either be done automatically by Bigtable or explicitly by client applications. To avoid collisions with explicit assignments, client applications must generate unique timestamps themselves.

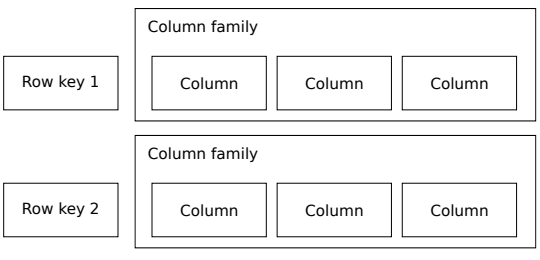


Figure 2. A visualisation of the data model used in Bigtable.

Cassandra is a distributed multi-dimensional map indexed by a row key and column name [14]; making it similar to Bigtable. Unlike Bigtable, Cassandra offers two kinds of columns families; so called Simple and Super column families. Simple column families are *regular* column families, but Super column families allow column families within column families as illustrated in Figure 3. Cells can be accessed by specifying the row key, column family name, and column name. A Super column family adds an extra level to the hierarchy and requires an application to specify the row key, column family name, super column name, and the column name. The row key is a string with no size restrictions. The column names can be ordered either by time or by name.

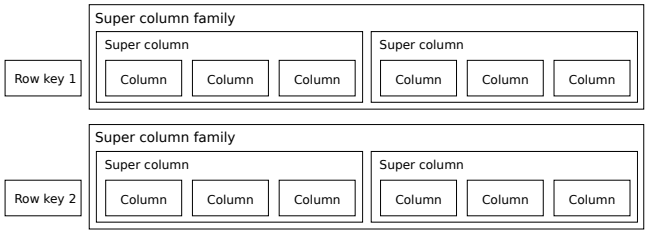


Figure 3. A visualisation of super column families as used in Cassandra.

In this section we described the data models of Dynamo, Bigtable and Cassandra. The next paragraph will give a brief summary.

As discussed Dynamo is not a storage system by itself, but uses other storage systems to store its data. However it hides these systems from clients by providing a simple interaction layer. Bigtable and Cassandra however do offer their own storage system. A commonality between these two systems is the use of column families in their data model. Cassandra improves the data model of Bigtable by adding super column families which consist of super columns. This addition allows for more complex data structures to be stored in Cassandra. An advantage of the pluggable persistence component which Dynamo offers is the ability to use databases which have proven themselves. It should be noted, however, that these systems are used in a very simple fashion since making full use of SQL databases, as stated in the beginning of this section, is inefficient. The simple data model Dynamo uses is powerful but has limited applicability.

Because Bigtable and Cassandra both use column families these systems are more flexible when it comes to storing data structures. With the addition of super column families by Cassandra, it offers the most features to be usable in different situations. However there are limitations in the way data is retrieved from the storage system, these limitations should be taken into account when designing a storage model.

4 PRACTICAL USE

A good way to get insight in the described systems is looking into the way they are used in practice. Bigtable and Dynamo are proprietary systems and their exact usage is not known publicly, however the papers describing these systems do mention some ways in which they are or could be used. Cassandra is open source and a number of companies have opted to use it instead of standard Structured Query Language (SQL) databases, these companies and the way they use Cassandra is described.

4.1 Dynamo

Dynamo is used by Amazon, a big online retail operation. For many services, such as those that provide best seller lists, shopping carts, customer preferences, session management, sales ranks, and product catalogs, Dynamo is used as a back end system. There is little information available on the exact usage of Dynamo but in the paper [6] a few general usage modes are described:

- *Business logic specific reconciliation:* In this usage mode data objects are replicated across multiple nodes. When versions of these objects differ, the client application is required to do its own reconciliation, i.e., choosing the right version. An example of a service within Amazon which uses this usage mode is the shopping cart service.

- *Timestamp based reconciliation:* When this usage mode is selected Dynamo performs timestamp based reconciliation where the last write wins, i.e., when two data objects conflict, the data object with the largest timestamp value is chosen. This usage mode is used in the service that maintains sessions of customers.

- *High performance read engine:* Some services have a high request rate and a low update rate. These services can change the quorum characteristics of Dynamo which allows them to use Dynamo as a high performance read engine. The high performance on reads can be achieved by setting R (minimal number of nodes that must participate in a successful read operation) to 1 and W (minimal number of nodes that must participate in a successful write operation) to N (total number of nodes). Services that provide product catalogues usually use this usage mode.

4.2 Bigtable

Google uses Bigtable in a number of different services, they describe exemplary use of Bigtable in some of their systems in their paper [4].

"Google Analytics is a service that helps webmasters analyze traffic patterns at their web sites. It uses Bigtable in a two table configuration; a raw click table and a summary table. The raw click table maintains a row for each end-user session. The row name is a tuple containing the name of the website and the time at which the session was created. This schema ensures that sessions which visit the same web site are contiguous and makes sure they are sorted chronologically. The summary table contains various predefined summaries for each website and is generated from the raw click table by periodically scheduled MapReduce jobs [5]. Each MapReduce job extracts recent session data from the raw click table.

Google also offers various services that display geographic image data, such as Google Earth and Google Maps. These services use one table for image data and a different set of tables to serve client data. Each row in the image table corresponds to a single geographic segment and the rows are named to ensure that adjacent geographic segments are stored near each other. The table contains a column family to keep track of the sources of data for each segment. This column family has a large number of columns; essentially one for each raw data image. Since each segment is only built from a few images, this column family is very sparse.

Personalized Search stores the data of each user in Bigtable. Each user has a unique *userid* and is assigned a row named by that userid. All user actions are stored in a table and a separate column family is reserved for each type of action (for example, there is a column family that stores all web queries). Each data element uses as its Bigtable timestamp the time at which the corresponding user action occurred. Personalized Search generates user profiles using MapReduce over Bigtable. These user profiles are used to personalize live search results. The Personalized Search data is replicated across several Bigtable clusters to increase availability and to reduce latency due to distance from clients. It uses a replication subsystem that is built into the servers." [2]

4.3 Cassandra

Cassandra was originally designed by Facebook to power their inbox search. Inbox Search was launched in June of 2008 for around 100 million users which grew to over 250 million users in 2009 [14]. The use of Cassandra for this purpose was abandoned in late 2010 when a new system for messaging was introduced based on Apache HBase [16].

Cassandra was made open source in 2008 by Facebook, development of this version has since been taken over by the Apache Software Foundation. It is still actively developed and has been adopted by several companies who use it to power different services. The following list gives a short description of these companies and how they use Cassandra.

- *Digg [22]:* A site for story sharing (usually through links) uses Cassandra as their main storage system [9, 20]. When a user *diggs* a story this is made visible to all his friends. Digg does this by selecting the *followers* of the user and adding the item to the list of items of each follower.

[2] From *Bigtable: A Distributed Storage System for Structured Data* [4]

- *Reddit [1]:* A site similar to Digg, they also use Cassandra as their main storage system [21, 12].

- *Twitter [8]:* A site which is popular for allowing people to post *tweets*; text-based posts of up to 140 characters. Twitter uses Cassandra for storing places of interest (geolocation) and the results of data mining done over their entire user base (research). Cassandra is not used as the storage system for tweets [13].

- *Cloudkick:* Provides monitoring tools for companies which have a large number of servers. They use Cassandra to store the usage data used in their tools [19].

These sites all have high amounts of visitors each day and large amounts of growing data to store. Visitors are encouraged to interact with the data and in the case of Digg and Reddit, users can tell if they like or dislike a certain item. This requires a storage system that allows a large number of fast reads and writes.

5 Discussion

In this paper we discussed three NoSQL distributed storage systems (DSS); Dynamo, Bigtable and Cassandra.

Dynamo, designed by Amazon, is a proprietary highly available key value-store. It uses nodes which are structured in a ring network for storing (replicated) data. It uses a very simple data model which is analogues to a hash map, a single key maps to a single value. The data itself is stored in an existing database through the use of its pluggable persistence component. It offers three parameters which allows the system to be tuned.

Bigtable is a proprietary distributed storage system designed by Google. It uses a client/master network structure in which data is stored by so called tablet servers, a master server makes sure single tablet servers are correctly load balanced. Bigtable provides a data model which consist of column families which contain columns and are stored in a row. Column families are predefined but it is possible to have multiple column families within a tablet, e.g. multiple rows use different column families. By using timestamps it is also possible to have multiple versions of the same data item.

Cassandra was designed by Facebook and has been made open source and is available for public use. Similar to Dynamo it makes use of nodes which are structured in a ring network. Its data model, however, is similar to Bigtable as it also provides column families but extends this notion with Super column families which stores multiple column families. It is also possible to add columns to column families without predefining them.

In section 2 we discussed the system architectures of the three NoSQL DSS. We said Bigtable has the weakest system architecture design with respect to its network structure and replication capabilities due to its use of the client/master structure. This makes Bigtable vulnerable to failure of its master server which makes the system insufficient over time. Because Dynamo and Cassandra use a preference list and can replicate data at multiple nodes, they do not suffer from single or, in some cases, multiple node failure. Therefore it is our opinion this is the stronger network structure.

The data models of the three NoSQL DSS were discussed in section 3. Dynamo has a simplistic data model and does not offer data to be structured, clients can add and parse structures if needed. Bigtable and Cassandra add more complexity by adding column families which allows the data to be structured. Cassandra extends the complexity by offering data to be modelled with Super column families, adding more structuring capabilities to the data model. It is clear Cassandra provides the most complex data model which allows data to be structured more easily. However there are limitations in the way data can be retrieved, these limitations should be taken into account when designing a system that uses Cassandra.

One aspect which is not addressed in the papers about Dynamo, Bigtable and Cassandra is security. In the Dynamo paper it is assumed that the environment in which Dynamo operates is non-hostile, implying it does not need security measures. Although the Bigtable paper does not mention security, we can assume a similar argument can be

applied because the environments are similar. The Cassandra paper also does not mention security, however Cassandra can be run in hostile environments. If security is required, the user needs to resort to 3rd party solutions. We feel this is a shortcoming of Cassandra.

6 Conclusion

From our research we conclude that important aspects of well-designed NoSQL distributed storage systems (DSS) are;

- Use multiple nodes which are equal to each other for;
  - Easy replication
  - No single point of failure
- Have a sufficiently powerful data model without compromising speed
- Be open source (having an active community is desirable).

As mentioned in section 5 security was not considered. Security considerations of NoSQL DSS can be a topic for further research.

In our research we only focused on one open source NoSQL DSS. In future research a comparison between multiple open source NoSQL DSS might be interesting.

As mentioned in sections 3 and 5 care should be taken when designing a data model for Cassandra since there are limitations to the way data can be retrieved. There is little information known on how to design these data models, so we suggest it as a topic for a future research project.

Acknowledgements

The authors wish to thank Dr. Alexander Lazovik for his suggestions and review of this paper. They also wish to thank Amirhosein Shantia for his review.

References

[1] Advance Publications Inc. reddit. Website. http://www.reddit.com/.

[2] M. Burrows. The chubby lock service for loosely-coupled distributed systems. In *7th USENIX Symposium on Operating Systems Design and Implementation*, OSDI '06, Nov. 2006.

[3] T. Chandra, R. Griesemer, and J. Redstone. Paxos made live - an engineering perspective. *26th ACM Symposium on Principles of Distributed Computing*, June 2007.

[4] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A distributed storage system for structured data. In *7th USENIX Symposium on Operating Systems Design and Implementation*, OSDI '06, Nov. 2006.

[5] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *6th Symposium on Operating Systems Design and Implementation*, OSDI '04, Dec. 2004.

[6] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: Amazon's highly available key-value store. In *Proceedings of the 21th ACM Symposium on Operating System Principles*, Oct. 2007.

[7] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, PODC '87, 1987.

[8] J. Dorsey, E. Williams, and B. Stone. Twitter. Website. https://www.twitter.com/.

[9] I. Eure. Looking to the future with cassandra. Website. http://about.digg.com/blog/looking-future-cassandra.

[10] I. Gupta, T. D. Chandra, and G. S. Goldszmidt. On scalable and efficient distributed failure detectors. In *Proceedings of the 20th Annual ACM Symposium on Principles of Distributed Computing*, PODC '01, 2001.

[11] P. Hunt, M. Konar, B. Reed, and F. P. Junqueira. Zookeeper: Wait-free coordination for internet-scale systems. In *Proceedings of the 2010 USENIX conference*, USENIXATC '10, 2010.

[12] D. King. She who entangles men. Website. http://blog.reddit.com/2010/03/she-who-entangles-men.html.

[13] R. King. Cassandra at twitter today. Website. http://engineering.twitter.com/2010/07/cassandra-at-twitter-today.html.

[14] A. Lakshman and P. Malik. Cassandra - a decentralized structured storage system. In *3rd ACM SIGOPS International Workshop on Large Scale Distributed Systems and Middleware*, Oct. 2009.

[15] L. Lamport. The part-time parliament. *ACM Transactions on Computer Systems 16*, May 2007.

[16] K. Muthukkaruppan. The underlying technology of messages. Website. https://www.facebook.com/note.php?note_id=454991608919.

[17] MySQL AB and Oracle. Mysql. Website. http://www.mysql.com/.

[18] Oracle. Oracle berkeley db. Website. http://www.oracle.com/us/products/database/berkeley-db/index.html.

[19] A. Polvi, D. D. Spaltro, and L. Welliver. Cloudkick. Website. https://www.cloudkick.com/.

[20] J. Quinn. Saying yes to nosql going steady with cassandra. Website. http://about.digg.com/node/564.

[21] Reddit admins. Reddits may 2010 state of the servers report. Website. http://blog.reddit.com/2010/05/reddits-may-2010-state-of-servers.html.

[22] K. Rose. Digg. Website. http://digg.com/.

[23] Wikipedia. Cartesian product. Website. http://en.wikipedia.org/wiki/Cartesian_product.