Self-Organizing Maps Applied in Visualising Large Software Collections

James Brittle and Cornelia Boldyreff
Department Of Computer Science
University Of Durham
{*j.g.brittle,cornelia.boldyreff*}*@durham.ac.uk*

Abstract

The self-organizing map's unsupervised clustering method can be used as a data visualisation technique. Within this paper different techniques to visualise self-organizing maps (SOM) and their effectiveness are investigated in relation to the organisation of a large software collection and its visualisation.

GENISOM, an offspring component of the GENESIS software engineering platform, incorporates the generation, maintenance and viewing of Self-Organizing Maps.

The results from our studies indicate that a hybrid of 2D and 3D visualisations is favoured by users. Extensive usability tests also show that the majority of users found that the additional information a SOM provides, aids browsing and searching of a software collection. Further work is addressing the problems found in the application of SOM within a software engineering environment.

Key Words: Self Organizing Maps, GENISOM, software visualisation

1 Introduction

Interactive exploration of a large software collection or large software systems, where the user looks at individual artefacts one at a time would be greatly aided by ordering them according to their contents.

There exist many possibilities to achieve this organisation, e.g. as a graph or a hierarchical structure. A common organisation is one in which the artefacts are represented by points on a 2D plane and the geometric relations between them relate to their similarity. Such representations are often called document maps. These organised collections of data facilitate a new dimension in information retrieval namely the possibility to locate pieces of relevant or similar information that the user was not explicitly looking for.

Document maps can be constructed through a number of methods including the data visualisation technique, the Self Organizing Map (SOM), invented by Prof. Teuvo Kohonen in the early 1980s [7]. SOMs are an unsupervised, clustering algorithm, which use neural networks. They have been demonstrated to aid programmers in the process of reverse engineering by discovering common features within legacy code [2] and to assist in object recovery[1], although visualisation of the associated maps is not explicitly considered in reports of this research.

A prominent problem within the field of Software Engineering concerns reuse. Reusable assets are in abundance, over the web and in libraries but it is extremely difficult to locate reusable software artefacts that are relevant to a particular application. The necessary organisation is often lacking and difficult to achieve given the dynamic nature of such software collections. This problem can also be found where a large evolving software system consists of an ever growing number of components and the managment and hence the comprehension of the associated software artefacts tends to be increasingly difficult.

Having suitable visualisations of such software collections can mitigate the problem identified above. The application of information visualisation builds upon the strenghts of humans and computers as

> "by properly taking advantage of peoples' abilities to deal with visual presentations, we may revolutionise the way we understand large amounts of data" [5]

Within this paper the use of Self Organising Maps as a means of visually presenting large software collections is demonstrated. Section 2 describes the implementation of GENISOM. Section 3 details the different visualisations the software can produce. Section 4 presents the results of the evaluation of the tool and subsequent adaption of maps. Section 5 identifies further work.

2 GENISOM

GENISOM is a client/server application designed to manage and enable viewing of SOMs. Figure 1 illustrates a simplified architecture diagram for the GENISOM system.

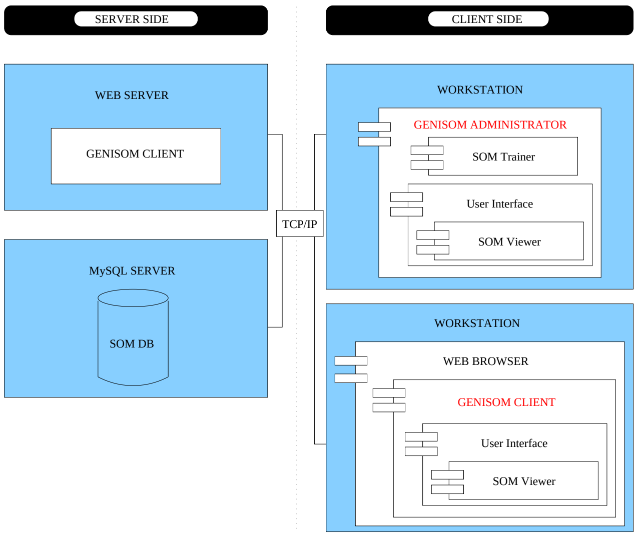

Figure 1. GENISOM architecture

The GENISOM Administrator is used to manage SOMs; it enables their creation and maintenance. The generated SOMs are then stored within a MySQL database, from which the GENISOM Client can then retrieve the data and display the generated maps. The Client is web based using Java Web Start[1] to aid its accessibility to users. The architecture of the system enables distributed software engineering teams to work together (see Figure 2).

Input data to create a SOM can be descriptions of any content as long as it is stored within a database table. For example each record could be a description of a reusable artefact or of a software component within a large software sytem such as a method in a class.

There are two main use case scenarios for the system; firstly the Administrator could be used by the librarian of a reuse library and the Client by the system developers (see Figure 3), therefore aiding them in the location of reuse candidates. Secondly both tools could be used by members of a software development team to aid program comprehension, or help in decisions regarding restructuring and reengineering of the system.

3 Visualisations of Maps

The GUI for the SOM evolved through a number of stages, due to the interactive design approach adopted.

[1]http://www.java.sun.com/products/javawebstart/

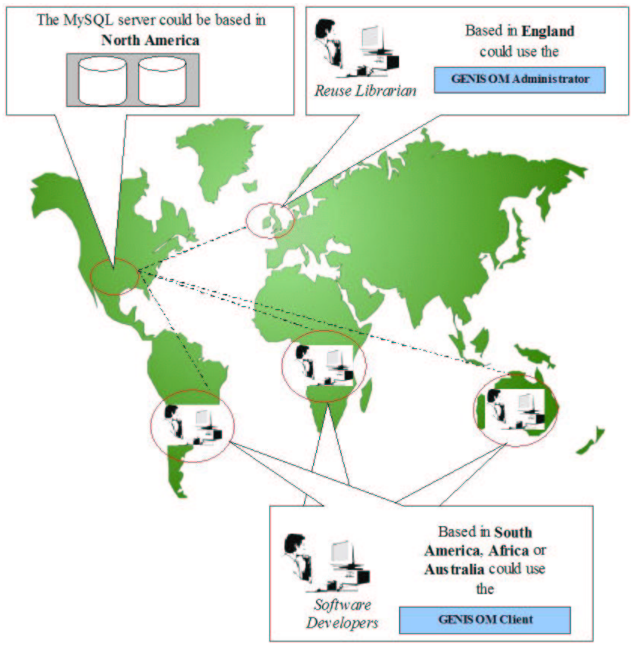
The MySQL server could be based in North America
Reuse Librarian
Based in England could use the
GENISOM Administrator
Software Developers
Based in South America, Africa or Australia could use the
GENISOM Client

Figure 2. Possible geographic layout of GENISOM system

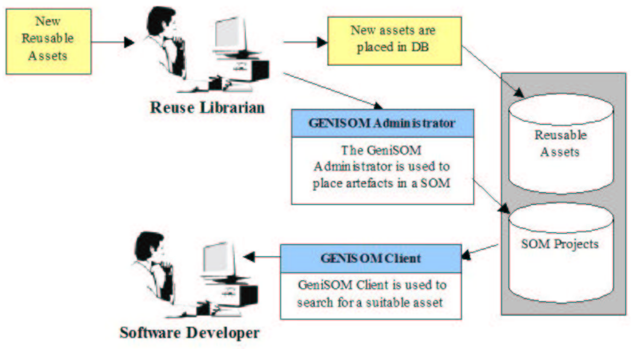
New Reusable Assets
Reuse Librarian
New assets are placed in DB
GENISOM Administrator
The GeniSOM Administrator is used to place artefacts in a SOM
Reusable Assets
SOM Projects
GENISOM Client
GeniSOM Client is used to search for a suitable asset
Software Developer

Figure 3. Reuse infrastructure with GENISOM system in place

Earlier designs drew inspiration from and were similar to WEBSOM [8], a web based example of the use of SOMs for organisation of document collections. Feedback through quick and dirty evaluations [12] was critical of the initial colour scheme used. This resulted in the final 2D map as illustrated in Figure 4. The improved colour scheme is based upon the use of the primary colours, essentially to help distinguish different elements of the map more clearly than the heatmap approach applied in WEBSOM.

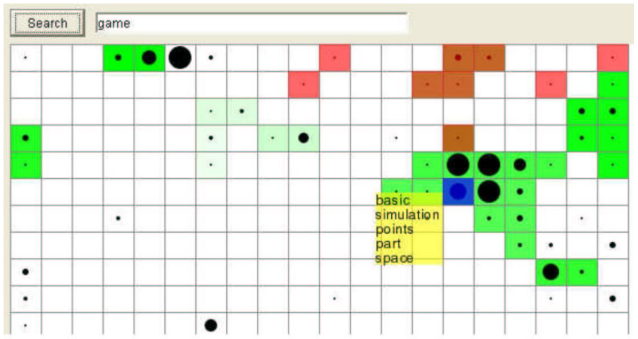

Figure 4. Screenshot of 2D Map

The neural net is arranged as a grid with the inputs (e.g. reusable artefacts) being attached to the neurons. The black dots on grid cells (i.e. neurons) indicate that inputs have been matched to them, with the size of the dot representing the number of them. The green shading of the grid cells indicates the boundaries of similar clusters of neurons.

The map is interactive allowing the user to select a particular neuron (the blue cursor indicating the selection), this displays the neuron's labels, five or less words that best describe the inputs matched to it (see Figure 5). This labelling approach, taken from LabelSOM [13], is a commonly used method within SOM software. As well as displaying the labels for a selected neuron, details of the inputs matched to it are also displayed in a side bar though this is not depicted in the figures.

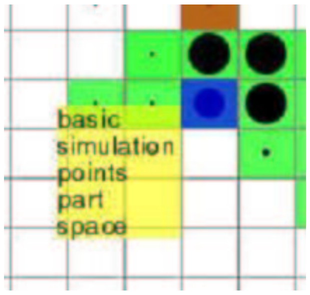

Figure 5. Closeup screenshot of 2D Map

Browsing is aided by a search system coupled to the map which highlights the results in red for a certain search string.

Using 2D limits the amount of information that can be visualised. Improvements to the GENISOM GUI therefore naturally led on to the development of a 3D map using the Cityscapes technique which has already found application in software visualisation [6]. Using this technique in GENISOM each value is plotted as a column (or 'building'). The 'buildings' are plotted on the same horizontal plane, allowing differences in height and position to be analysed. In relation to a SOM, each building represents a neuron and the height of the building relates to the number of inputs that are matched to it. The implementation of this used Java3D[2] and in the final system the option was made available to switch between using 2D and 3D interfaces.

Figure 6 illustrates the 3D map; the user's view can be rotated and zoomed in and out. Furthermore, the user can also interact with the 3D map in the same manner as the 2D map following the same colour scheme (see Figure 7).

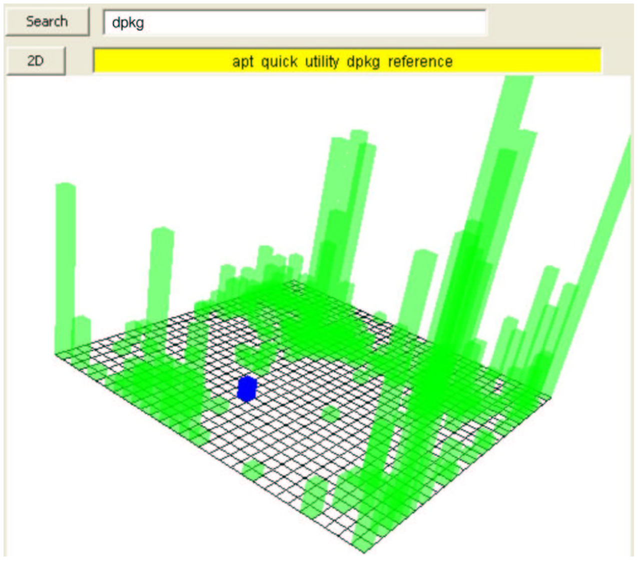

Figure 6. Screenshot of 3D Map

4 Evaluation and Results

A series of different evaluations were carried out to assess the success of the GENISOM software. As test input data, a selection of 300 Debian Packages were gathered from the Debian Project. This is an open source development of a free OS, Debian GNU/Linux. New contributors to the project as well as users interested in studying the Debian project could be the potential beneficiaries of these maps. In the evaluations all participants were from the Department of Computer Science at the University of Durham.

[2]http://java.sun.com/products/java-media/3D/

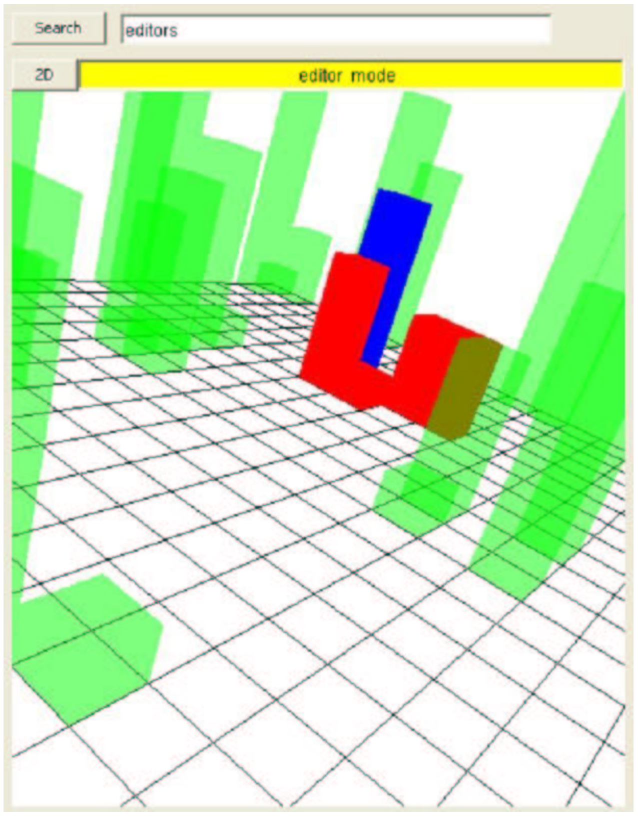

Figure 7. Screenshot of 3D Map

4.1 Administrator

The Administrator was assessed for usability as it is a critical area of the software, in that the user should be able to generate maps proficiently. Results from a heuristic evaluation showed that the technical terminology used within the software and also the task of assessing the ’goodness’ of the SOM produced could be problematic. The former was found to be the case after carrying out usability testing and questionaires. The anticipated problem of assessing the ’goodness’ of a map did not actually occurr in practice during the evaluations.

Overall the Administrator has a steep learning curve, which is to be expected with such a technical piece of software. However it is predicted that once the terminology (i.e. language) barrier has been overcome, operation of the system can be quickly mastered with training. Trials carried out with software engineering researchers proved this to be the case.

4.2 Client - 2D versus 3D

Secondly the Client was evaluated, in two distinct trials. The first of these involved a comparison between the two types of interface with respect to usability and efficiency. A population of 10 computer scientists carried out a series of usability exercises, which involved searching for information held within the map.

Statistical results from these evaluations indicated that the 2D interface was superior to the 3D. The 2D was more efficient with respect to the time required to complete the tasks and more effective due to the observed signs of frustration shown by the users. All participants considered that the Cityscape technique did inform them far better on the spread of the population (i.e. the heights of the buildings were more intuitive indicators of the population density than the different sizes of black dots), however this feature was not considered a necessity. Overall an overwhelming majority, 90%, preferred to use the 2D interface.

Reasons for their opinions mostly arose from the difficulty in selecting a ’building’ (i.e. a neuron) in the Cityscape, as certain buildings could obscure others meaning that the view of the scene would have to be changed to enable selection of the obscured buildings. Other comments noted the poor navigation through the 3D scene, which is a common problem with such applications. It is difficult to control the 3D space with interaction techniques that are currently in common use since they were designed for 2D manipulation (e.g. dragging and scrolling) [11].

Overall conclusions drawn from the evaluation were that the combination of the two maps actually complemented each other. The 2D map was quick and simple to use; however, the 3D map did add functionality to the browsing which concurs with the findings of Koua and Kraak [9].

4.3 Client - Major Evaluation

In the second part of the evaluation of the Client, a key question sought to determine whether using SOMs was an improvement against other methods. This led to the creation of the GENISOM Testbed, an application that could be downloaded by participants and would essentially guide them through an evaluation while automatically logging the results.

The Testbed consisted of two searching tasks using the 2D map and as a comparison two tasks using a standard search system, similar to the functionality of the Google web search engine. It also included a questionnaire to record the users' experiences.

114 participants took part in the evaluation, on the basis of the statistical results the 2D map proved to be less effective, as it took users longer to complete set tasks with the 2D map. User opinions on the effectiveness of the new system were mixed though; over 30% thought that it was more effective and only 52% stated it was less effective. Overall, however, the evaluation statistically regarding the times to complete tasks was seen to be flawed due to the difference in learning curves of the two systems. The 2D map provides a fairly complex interface and therefore has a steep learning curve, while users will already have obtained most knowledge of how to operate the standard search system due to prior experience with web search engines.

Predictions regarding future fair usability tests are promising for the SOM as users were found to vastly reduce their times for the second task using the map even though the second task was considerably more difficult. Also participants' views on whether the additional information the SOM provided, aided searching and browsing showed that over 68% thought this to be true, which concurs with Merkl's findings:

"such a library system has its benefits when developers look for a particular component during the development process of a software system. Additionally the self-organizing map can be used easily for interactive exploration of the software library - a feature that is of vital importance in software reuse"[10]

Although Merkl's paper is very positive towards the application of Self-Organizing Maps to reuse libraries, its results were subjective. The asset population was small in comparison to what would be found in a normal reuse library and there was no formal evaluation of the prototype system developed.

4.4 SOM Algorithm

Evaluation of the algorithm behind the Administrator consisted of creating a number of different maps with varying input size, 25 to 13000 Debian Packages. The very nature of the standard SOM algorithm is that a number of trial maps need to be created to achieve the best map:

"an appreciable number (say, several dozens) of random initialisations...and different learning sequences ought to be tried, and the map with the minimum quantization error selected."[7]

The results from the evaluation back up this theory, however, they present a possible problem for the use of SOMs in this application. Generating maps is a time consuming activity due to the memory latencies of self-organizing neural networks. Coupled with the factor that many maps will need to be trialled the feasabilty of using the standard SOM algorithm in a software engineering environment is fairly low, unless the software collection is stable and justifies the initial investment in the map production. Further work into researching more advanced algorithms is therefore a main priority of our work.

4.5 Outstanding Problem

Feedback highlighted one major criticism of both the GENISOM 2D and 3D map interfaces. The criticism was that initially the view of the maps was 'uninformative' as information, i.e. grid cell label, was only displayed once the cell had been selected. Therefore the user must systematically select cells to build up a mental model of the map's contents. However, displaying all the labels at the same time led to a cluttered map therefore it was decided let the user have control over which label was displayed. A better mechanism for providing the user with an overview of the map's contents other than exposing the textual labels over the whole map is being sought. Currently structuring the SOM hierarchically is being investigated.

5 Conclusions and Further Developments

The evaluation of GENISOM highlighted many different areas of improvement. Firstly regarding the SOM algorithm, a hybrid algorithm the Growing Hierarchical Self-Organizing Map (GHSOM)[3] is proposed as a replacement. This has a number of benefits: the training process is virtually automated minimizing the number of parameters and in theory allowing the best map to be generated on the first attempt. This also has the advantage of simplifying the generation of the maps, which was found to be a troublesome area within the Administrator.

The GHSOM also enables a hierarchical structure of SOM to be built up, firstly this would aid visualisation of the information in respect to the SOM's organisation and the earlier stated problem regarding labelling of the map. Secondly if actual source code was used as input data then it would facilitate object finding as demonstrated by Chan and Spracklen [2].

Regarding the different visualisations of SOM, the presented results suggest a combination of the 2D and 3D maps. Research into combining the two would be worthy of further consideration. Investigating other 3D techniques to visualise SOMs may lead to conclusions on whether the technique of Virtual Data Mining is applicable in this application.

The integration of the software to the GENESIS platform [4] is also proposed, giving scope for real life trials of the software and actual use by software engineering professionals. Investigations could also be carried out into the applicability of the SOM not just to software components but to a variety of reusable artefacts: test cases, designs and documentation. This would allow the possibility of further work following the findings of Chan and Spracklen to investigate in more detail the software analysis properties of the SOM.

References

[1] A. Chan and T. Spracken. Discovering common features in software code using self-organizing maps. In *Proceedings of the International Symposium on Computational Intelligence*, Kosice Slovakia, August 2000.

[2] A. Chan and T. Spracklen. Object recovery using hierarchical self-organizing maps. In *Proceedings of the International Conference on Engineering Applications of Neural Networks*, Kingston Upon Thames UK, July 2000.

[3] M. Dittenbach, D. Merkl, and A. Rauber. The growing hierarchical self-organizing map. In *International Joint Conference on Neural Networks*, volume 24, Como, Italy, July 2000.

[4] M. Gaeta and P. Ritrovato. Generalised environment for process management in cooperative software engineering. In *International Computer Software and Applications Conference*, volume 26, pages 1049–1059, Oxford, England, August 2002. IEEE.

[5] C. Glymour, D. Madigan, D. Pregibon, and P. Smyth. Statistical themes and lessons for data mining. In *Data Mining and Knowledge Discovery*. Kluwer Academic Publishers, 1997.

[6] C. Knight. *Virtual Software In Reality*. PhD thesis, Durham University, 2000.

[7] T. Kohonen. *Self-Organizing Maps*. Information Sciences. Springer, second edition, 1997.

[8] T. Kohonen, S. Kaski, K. Lagus, J. Salojarvi, J. Honkela, V. Paatero, and A. Saarela. Self organization of a massive document collection. In *IEEE Transactions on Neural Networks*, volume 11, pages 574–585, May 2000.

[9] E. Koua and M. Kraak. An evaluation of self-organizing map spatial representation and visualization for geospatial data: Perception and visual analysis. Technical report, International Institute for Geo-Information Science and Earth Observation (ITC), 2001.

[10] D. Merkl. Self-organizing maps and software reuse. In *Computational Intelligence in Software Engineering*. World Scientific, 1998.

[11] J. Nielsen. 2d is better than 3d. useit.com - Jakob Nielsen's Alertbox, 1998.

[12] J. Preece, Y. Rogers, and H. Sharp. *Interactive Design: beyond human-computer interaction*. John Wiley and Sons, 2002.

[13] A. Rauber. Labelsom : On the labeling of self-organizing maps. In *International Joint Conference on Neural Networks*, 1999.