A Pattern-Based Approach to Support Automatic Homogeneous Map Labeling with Texts, Charts and Other Elements in a WMS

Enrique Andreu, Rubén Béjar, Miguel Á. Latre, Sergio Martínez, Pedro R. Muro-Medrano
Department of Computer Science and Systems Engineering, University of Zaragoza
E-50018 Zaragoza, Spain
{enriquea, rbejar, latre, sergiom, prmuro} @unizar.es

SUMMARY
This paper offers two contributions to the map labeling problem: first of all the "generic label" analysis pattern is defined, in order to allow the homogenous treatment, in a software with GIS visualization and/or printing capabilities, of text labels, statistical charts, graphic icons and other elements that have information about the features on a map and are drawn on top of it. The second contribution is an approach, based on simulated annealing, to efficiently support map labeling with generic labels. Some results, examples of different maps labeled and their rendering times, will be presented to show the viability of this approach implemented in a Web Map Service.

KEYWORDS: *Map Labeling, Software Analysis Pattern, Web Map Service*

INTRODUCTION

Text labels are an essential feature that makes a map readable and informative by naming the features displayed on it. When the need arise to provide more information on a map, as in thematic maps, some classical solutions exist, as choropleth maps that use colors to represent spatially distributed variables. If even more complexity is needed, i.e. taking into consideration several variables at the same time, statistical charts are often used; they provide a way to compare these spatially distributed variables by just having a look at the map.

This paper proposes to manage homogeneously text, charts, icons, etc., by means of a generic labeling solution. Although many research address the map labeling problem, most of it is specifically for texts or doesn't consider its application to a variety of elements in maps. For example a recent work by (Kreveld, Schramm & Wolff, 2004) studies specifically the placement of statistical charts in areas, as a special case of the map labeling problem; with the abstraction for labels proposed in this paper, many of their findings would also be applicable to texts. And when the positioning of texts and other elements is beheld, they are treated as different elements (Harrie, Zhang & Ringberg, 2005). This work offers a solution to label maps composed by point, line and polygon features, efficiently and with a common approach, structured and defined as a software analysis pattern, for elements that previously could have not been treated uniformly as labels (i.e. statistical charts). The solution has been implemented in a Java Web Map Service and is efficient enough for its daily use. It is described in detail in the next section.

The algorithmic part of the map labeling solution proposed here is based in a simulated annealing algorithm as described in (Edmonson et al., 1996), though this paper applies it only to texts. This choice requires algorithms for the generation of label candidates, that are candidate places to put labels by their features, and a labeling quality evaluation procedure. This is described in the third chapter of this work.

THE *GENERIC LABEL* PATTERN

Although the pattern concept has been used in software analysis, architecture, and design for some ten years, there is not a precise, commonly agreed definition. Martin Fowler gives a simple and generic definition in (Fowler, 1997): “A pattern is an idea that has been useful in one practical context and will probably be useful in others”. Fowler adds more regarding to analysis patterns: “[...] patterns that reflect conceptual structures of business process [...]”. The importance of patterns is that they allow to share reusable knowledge about architecture, analysis and design in a structured way. The rest of this paper assumes these definitions.

This chapter presents an analysis pattern we have called *Generic Label*. In this case, the business domain is GIS, and the pattern offers a solution to support flexible map labeling in GIS software. One important thing of patterns is that they should be discovered, not invented. The *Generic Label* pattern has been extracted from the second major refactoring and extension of the labeling infrastructure in a Java OGC Web Map Service in development in our laboratory since year 2000 (Fernández et al., 2000).

- **Context**: GIS applications with visualization and/or printing capabilities.
- **Problem**: There are many examples of situations that require automatic positioning of information (i.e. labels) about features on a map:
  - Automatic positioning of text labels to create cartographic-quality maps form geodata.
  - Solutions for thematic mapping that include statistical charts, icons etc. to increase visual information on the different features on a map.
  - Moving objects over maps, that require constant identification and status info, i.e. truck fleet tracking by GPS.

  These situations often happen where interactivity and efficiency is needed, i.e. Web mapping, and in very dynamic environments, where new geodata and geoservices appear, change and disappear frequently (i.e. in an SDI, where the different actors will be providing their geodata and geoservices and will have their own updating plans and paces).

  In this situations, is not trivial for a GIS software to provide solutions to build fast, dynamic maps, that still show enough information and have enough cartographic quality to be useful in a wide range of uses.

- **Solution**: The *Generic Label* pattern provides a generic concept that allows to separate **content**, i.e. feature attributes used to create a label, **position and size**, that change dynamically, and **symbolization**, i.e. as a text string.

  This separation allows to represent the same label content in different ways, like the Document-View design pattern does for the design of graphical interfaces (Buschmann et al., 1996), while offering support for an automatic map labeling algorithm that is independent of both label content and symbolization. This way new symbolizations and/or map labeling algorithms may be added independently without having to change the label content model.

- **Structure**: The UML class diagram that reflects the structure of the *Generic Label* pattern is shown in *Figure* 1. The three most important classes are:
  - **GenericLabel**: An element, with information about a feature, that is positioned over a map in a certain position and with a certain size (i.e. with a certain bounding polygon). This element will show the information in a LabelContent and will be drawn by a LabelRenderer. The Map Labeling Algorithm will only need to work with these GenericLabels, thus being independent from the content or the way to draw them.
  - **LabelContent**: The information about a feature that should be drawn on a map next to this feature, whenever it is visible, and following some cartographic criteria (i.e. appropriate size and style, avoiding overlapping with other similar elements, favoring certain positions next to its associated feature etc.). Several different LabelContents may be defined for a Feature.
  - **LabelRenderer**: It is the class that draws GenericLabels on a map. Extending it allows to have different symbols (i.e. texts, charts, icons) to render the LabelContents of these GenericLabels.
  - The other classes are given mainly for context, and are simplifications chosen to show how this pattern could be used in a context: A Map would be composed of Features, with their Attributes, and the user would associate LabelContents and LabelRenderers to every Feature. The Map would create GenericLabels based on these LabelContents and LabelRenderers (it needs both to calculate the size and position of the GenericLabel) and would give them to the map labeling algorithm to be positioned.

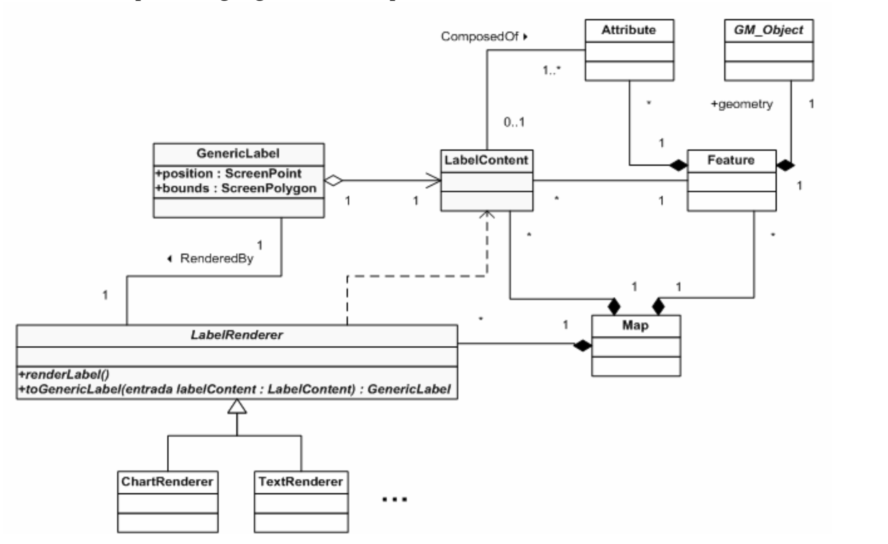


Figure 1: UML class diagram for the *Generic Label* pattern

- **Uses**: An example of map labeled by our software is given in *Figure* 2 to show this pattern in action. In this example, Features are some USA States, and the GenericLabels are created with two different LabelContents composed one of them by three Attributes (number of white, black and hispanic people in every State) and the other one by one Attribute (the name of the State). Three LabelRenderers are simultaneously used: one that shows a LabelContent as plain Text in size 10, other one that shows a LabelContent as bold Text with size 12 and finally one that renders its Content as a Pie Chart. All the GenericLabels are drawn completely inside of its corresponding State and do not overlap among them. This maps shows how the *Generic Label* pattern allows to automatically mix different kinds of LabelContents and LabelRenderers in the same map, while avoiding overlaps and following the cartographic criteria defined in the labeling algorithm

- **Consequences**: The application of the *Generic Label* pattern allows to integrate new kinds of labels easily, by extending the class LabelRenderer, avoiding changes in the other elements (without any changes in the map labeling algorithm or in the content of the labels). For example is a typical requirement in GPS fleet tracking applications to show the state of the different sensors mounted on board the vehicles (speed, state of the doors, alarms etc.). It would be nice to have the information of these sensors as icons that are shown around the representation of every vehicle. Thanks to the separation of label content, symbolization and position that provides this pattern, the solution to this would be extending LabelRenderer to provide an IconRenderer, specialized in painting these icons, and everything else in the system could remain unchanged. Another consequence is that it makes it possible for the map labeling algorithm to avoid overlapping among different elements drawn over the map, as the pattern provides the way for this algorithm to manage them all the same. Finally this pattern makes it trivial to support multiple labels, with different label renderers, for one feature.

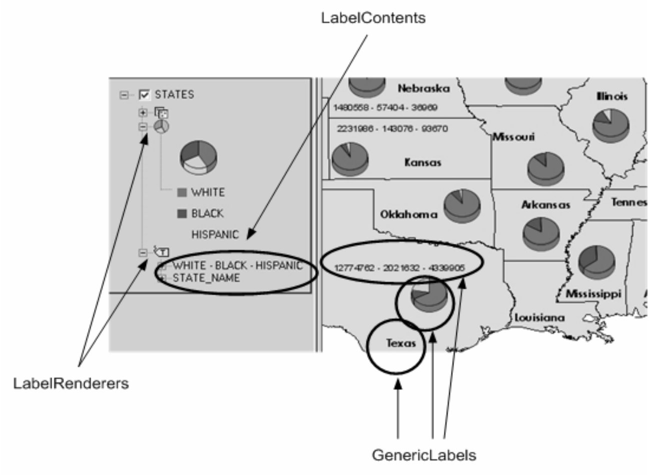


Figure 2: The Generic Label pattern in action

LABELING A MAP WITH GENERIC LABELS

Generic Labels offer a solution to manage different kinds of labels on a map, but they do not provide a solution for an automatic quality labeling. A map labeling algorithm, adapted to use generic labels, is still needed. Simulated annealing, as described in (Edmonson et al., 1996) was chosen as the base for this algorithm. It briefly consists in calculating several **candidates**, possible positions to place a label, for each feature to be labeled giving each candidate a preference level or weight, and then selecting one for each feature and **evaluating** the quality of the resulting labeling, repeating this latest steps until a good solution is found. Finally any remaining label overlaps are solved removing all but one of the overlapping labels.

Regarding both evaluation of the quality of a labeling, and the generation of candidates for the labels, the solution proposed here follows some common cartographic criteria to make readable and understandable maps with labels: a good classification of many of those label placement rules is given in (van Dijk et al., 2002).

The idea for the evaluation function used was also taken from (Edmonson et al., 1996), and briefly is a summation of all the weights of the selected candidates for each label in a given labeling, with a penalization for labels overlapping. A variation of the penalization for labels was tried, that depended on the percentage of overlapping among labels; nevertheless this idea resulted worse both in time and labeling quality.

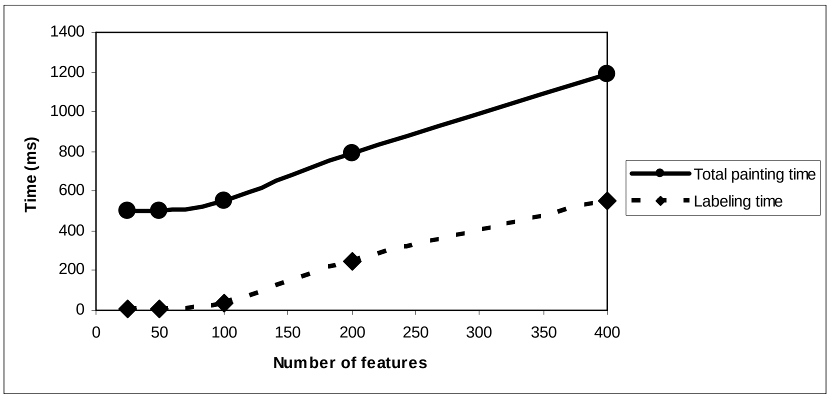
The candidate generator algorithms define the possible places where generic labels can be positioned by their features. All these generators work properly without needing information about content or symbolization of the labels, they only need their shape and the feature they are labeling: this allows to apply them to generic labels. Although the *Generic Label* pattern allows any kind of polygons to define the shape of labels, rectangles have been used for efficiency reasons. Different algorithms were used for point, line and polygon features:

- Points: as these features are the simplest ones, the basic solution proposed in (Edmonson et al., 1996) has been applied with small variations. It consists in selecting several candidates around the point, uniformly distributed, and giving them different weights depending on cartographic criteria. Although Edmonson et al. suggest using up to seventeen candidates, we discovered that eight candidates, in the feature "cardinal points" and adequately weighted, were enough to achieve good quality with less computational effort.
- Polygons: (Dörschlag, Petzold & Plümer, 2003) briefly describe an erosion algorithm to place objects in areas without violating their boundaries. Our solution is based on some of these ideas, with a few simplifications to improve its efficiency. The main difference is that our solution chooses candidates randomly from the points inside the eroded polygon, instead of selecting them from a skeleton of this polygon. This is more efficient, and the labeling quality is still good.
- Lines: As this is the most complex geometry to label, we took some ideas from the proposals for the evaluation of line candidates given in (Edmonson et al., 1996), but an algorithm to generate these candidates was developed, after discarding some more complex, and slower, approaches such as the proposed in (Wolff et al., 2001) that treats text characters in labels independently, allowing for better placement but taking more time. The algorithm developed is as follows:
    1. Take a segment of the line
    2. If the label fits in this segment, generate two candidates: one above the line and centered in the segment, and the other below it, also centered
    3. Else If the label fits in the next segment, generate candidates for this segment
    4. Else Take segments until the label fits an imaginary line between the first and the last of them, and generate candidates for this line
    5. If the line still has segments, take the next and go to 2.

Candidates for lines are oriented according to the segment where they are positioned and separated from the line until they do not overlap it. Horizontality, and aboveness are taken as quality parameters. Also candidates generated in points 2 and 3 of the algorithm are preferred, as they will typically be closer to the line.

RESULTS

In *Figure* some experimental results of the proposed labeling solution are shown. A line coverage has been painted at different scales, measuring the drawing time for them. Although these results are only a small example, our different tests show that the labeling time we get is in the same order of magnitude than the feature painting time, typically ranging from some milliseconds to a second or so, what is quite a good response time for our needs.



Labels considered	25	50	100	200	400
Labels painted	24	44	81	155	250
Labeling time	5	6	34	247	550
Total painting time	500	503	550	793,4	1190,6

Figure 3: Labeling time for line features

The *Figure* shows an example of map labelled with the proposed solution and served by our WMS. It includes line, point and polygon labels (blue, black, in a yellow rectangle) and also pie charts, with some numeric data from the polygons (municipalities in our region). Although the cartographic quality of this map is of course improvable, we consider it a good result for the painting times achieved.

CONCLUSIONS

This work has presented two contributions to the map labeling problem. First of all, a software analysis pattern named *Generic Label* has been described as a solution for GIS software that needs to label maps, composed by features, with elements that do not need to be specifically texts. This pattern abstracts the concept of label as [content + position and size + symbolization], allowing thus a great amount of flexibility to support new kinds of labels as well as different map labeling strategies.

The second contribution is a strategy for automatic map labeling with generic labels, aiming at efficiency and completeness (i.e. being able to label maps with any kind of features and fast)

while still keeping a good cartographic quality. It has been shown that many contributions to this problem in the bibliography can easily be adapted for its use with generic labels. It has also been described how some decisions have been taken, and a new candidate generator for line labeling has been developed, in order to fulfil the requirements of an efficient, and still good enough, map labeling algorithm.

Both contributions have been implemented and tested in a WMS developed in our laboratory and in current use in several SDI initiatives, including the Spain national SDI (IDEE).

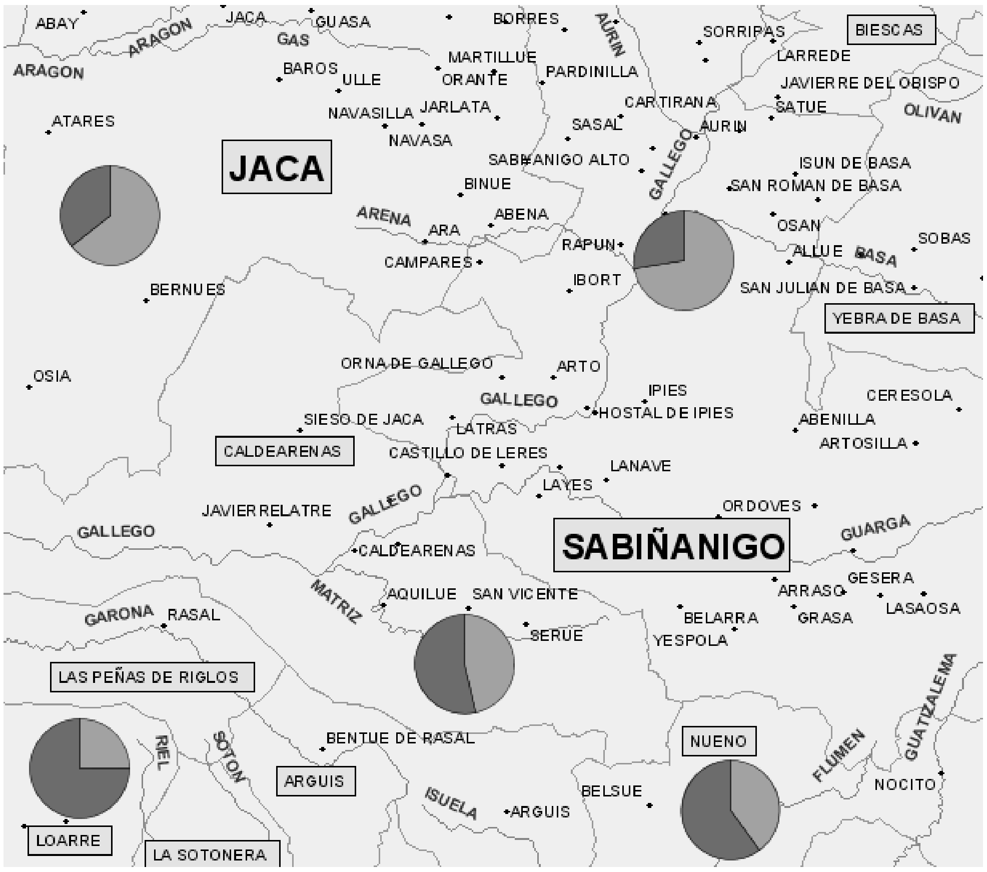


Figure 4: Example of map labelled with the solution proposed in this paper

ACKNOWLEDGEMENTS

This work has been partially supported by Zeta Amaltea S.L. and the Spanish Ministry of Education and Science through the project TIC2003-09365-C02-01 from the National Plan for Scientific Research, Development and Technology Innovation.

The basic technology of this work has been partially supported by Zeta Amaltea S.L. and the Spanish Ministry of Education and Science through the project TIC2003-09365-C02-01 from the National Plan for Scientific Research, Development and Technology Innovation.

BIBLIOGRAPHY

Buschmann F., Meunier R., Rohnert H., Sommerlad P., Stal M., 1996 A System of Patterns. John Wiley & Sons Ltd. ISBN 0471958697, pp. 140.

Dörschlag D., Petzold I., and Plümer L., 2003. Placing objects automatically in areas of maps. In Proc. 23rd International Cartographic Conference (ICC'03), Durban, South Africa, 2003

Edmondson S., Christensen J., Marks J., and Shieber S., 1997 A general cartographic labeling algorithm. Cartographica, 33(4) pp. 13-23

Fernández P., Béjar R., Latre M.Á., Valiño J., Bañares J.Á., Muro-Medrano, P.R. Web Mapping Interoperability in Practice, a Java Approach Guided by the OpenGis Web Map Server Interface Specification. Proccedings of the 6th European Commission GI and GIS Workshop, Lyon, France, 30 June 2000.

Fowler M., 1997 Analysis Patterns. Addison Wesley Longman Inc. ISBN 0201895420, pp. XV.

Harrie L., Zhang Q., Ringberg P., 2005 A case study of combined text and icon placement. Proceedings of the International Cartographic Conference 2005: Mapping Approaches into a Changing World, July 9-16, 2005, A Coruna, Spain, CD-ROM: Theme 12: Internet Location-Based Services, Mobile Mapping and Navigation Systems, Session 5.

van Dijk S., van Kreveld M., Strijk T., and Wolff A., 2002. Towards an evaluation of quality for names placement methods. International Journal of Geographical Information Science, 16(7) pp. 641-661

van Kreveld M., Schramm E., and Wolff A., 2004. Algorithms for the placement of diagrams on maps. In Dieter Pfoder, Isabel F. Cruz, and Marc Ronthaler, editors, Proc. 12th Int. Symp. ACM GIS (GIS'04), pp. 222-231

Wolff A., Knipping L., van Kreveld M., Strijk T., and Agarwal P. K., 2001. A simple and efficient algorithm for high-quality line labeling. Technical Report UU-CS-2001-44, Department of Computer Science, Utrecht University