Validation of the City Metaphor in Software Visualization

Gergő Balogh(✉)

Department of Software Engineering, University of Szeged, Szeged, Hungary
geryxyz@inf.u-szeged.hu

Abstract. The rapid developments in computer technology has made it possible to handle a large amount of data. New algorithms have been invented to process data and new ways have emerged to store their results.

However, the final recipients of these are still the users themselves, so we have to present the information in such a way that it can be easily understood. One of the many possibilities is to express that data in a graphical form. This conversion is called visualization. Various kinds of method exist, beginning with simple charts through compound curves and splines to complex three-dimensional scene rendering. However, they all have one point in common; namely, all of these methods use some underlying model, a language to express its content.

The improved performance of graphical units and processors have made it possible and the data-processing technologies have made it necessary to renew and to reinvent these visualization methods. In this study, we focus on the so-called city metaphor which represents information as buildings, districts, and streets.

Our main goal is to find a way to map the data to the entities in the fictional city. To allow the users to navigate freely in the artificial environment and to understand the meaning of the objects, we have to learn the difference between a realistic and an unrealistic city. To do this, we have to measure how similar it is to reality or the city-likeness of our virtual creations. Here, we present three computable metrics which express various features of a city. These metrics are compactness for measuring space consumption, connectivity for showing the low-level coherence among the buildings, and homogeneity for expressing the smoothness of the landscape. These metrics will be defined in a formal and an informal way and illustrated by examples. The connections among the high-level city-likeness and these low-level metrics will be analyzed. Our preliminary assumptions about these metrics will be compared to the opinions of users collected by an on-line survey. Lastly, we will summarize our results and propose a way to compute the city-likeness metric.

Keywords: Software visualization · City-metaphor · Validation · Metric

1 Introduction

In theory software systems could became infinitely complex by their nature. In theory, there is no limit of control flow embedding, or the number of methods,

O. Gervasi et al. (Eds.): ICCSA 2015, Part V, LNCS 9159, pp. 73–85, 2015.
DOI: 10.1007/978-3-319-21413-9_6

attributes, and other source code elements. In practice, these are limited by the computational power, time and storage capacity. To comprehend these systems, developers have to construct a detailed mental image. These images are gradually built during the implementation of the software system.

Often these mental images are realized as physical graphics with the aid of data visualization software. For example, different kinds of charts are used that emphasize the difference among various measurable quantities of the source code, or UML diagrams which are able to visualize complex relations and connections among various entities in the system.

There are several visualization techniques which use various real-life entities to represent abstract concepts. We used the city as a metaphor for the software system itself. To make it easier to navigate in a virtual environment and to help interpret the underlying connections and concepts, the generated world has to be similar to the real world. In our case it means we have to generate realistic cities to represent the abstract concepts and relations among the properties of the source code. To create such a city without manual intervention, we need a way to connect low-level properties of the city with its degree of realism.

1.1 Research Questions

Our brains are hard-wired to grasp the meaning of real objects. To make navigation easier in a virtual environment and to help to interpret the underlying connections and concepts for the user, the generated world has to be quite similar to the real world. In our case it means we have to generate realistic cities to represent the abstract concepts and relations among the properties of the source code. To create such a city without human intervention, we need a way to connect low-level properties of the city with its degree of realism. These problems can be summarized in the following research questions.

RQ. 1 Is it possible to define a set of low-level, directly measurable metrics of the generated city that is able to describe its properties in a meaningful way?
  - We will define a set of proposed metrics that fulfill these requirements.
  - To ensure the validity of these constructs, we will conduct a user survey.

RQ. 2 Is there a way to determine the degree of realism of the generated city based on low-level metrics?
  - We will look for a way to combine the low-level metrics in order to create a high-level metric that is able to express the similarity between a real city and a generated city.

2 Background

2.1 Related Works

Everyone is different it seems, each person having his or her own points of views. We use various tools to comprehend the world. Some of us need numbers, others use abstract formulas; but most of us need to see the information as colors, shapes or figures. To fulfill the expectations of users, many data visualization techniques and tools have been designed and implemented. It is beyond the scope of this article to exhaustedly evaluate these techniques and tools, but in our opinion traditional visualization tools like Rigi [6], sv3D [3] and SHriMP Views [4] were built on innovative ideas. It is often difficult to interact with them, and they usually "fall behind" in graphics terms compared to those in today's computer games, for instance. To address these issues we implemented a tool which combines today's common interactive, three dimensional game interface with an elaborate static code analysis.

Closely related approaches to our tool are CodeCity [5] and EvoSpace [2], which use the analogy of skyscrapers in a city. CodeCity simplifies the design of the buildings to a box with height, width, and color. The quantitative properties of the source code – called metrics – are represented by these attributes. In particular, each building represents a class where the height indicates the number of methods, the width represents the number of attributes, and the color tells us the type of the class. The buildings are grouped into districts as classes are tied together into namespaces. The diagram itself resembles a 3D bar chart with grouping. EvoSpace uses this analogy in a more sophisticated way. The buildings have two states: closed – when the user can see the large-scale properties like width and height, and open – when we are able to examine the small-scale structure of the classes, see the developers and their connections. It also provides visual entity tagging and quick navigation via the connections and there is also a small overview map.

2.2 Code Visualization in CodeMetropolis

Our tool called CodeMetropolis is a collection of command line tools. It takes the output graph of SourceMeter [1] and creates a Minecraft world from it. SourceMeter are a collection of various programs that are able to analyze and measure static artifacts related to the source code. The output is given with a unique binary format. The world uses the metropolis metaphor, which means that the source code metrics are represented by the various properties of the different building types. Figure 1 shows an example world.

In our representation there are two levels. The data level contains the various objects and their data, which are directly related to the measured artifacts, such as classes. However, the metaphor level is built up from the visual representations of these like buildings and floors. At the data level, each entity has its own property set – for example metrics, which are displayed at metaphor level. The buildings in our metropolis are parts of this metaphor, and they have some attributes which control the visual appearance. The items which are highlighted in Figure 2 represent various source code entities, while the properties are mapped to the attributes in order to visualize the data. In this concrete case, we can see some namespaces visualized as stone plates. They contain two classes represented by buildings. Lastly these have several methods whose size and complexity are mapped to the width and height of the floors.

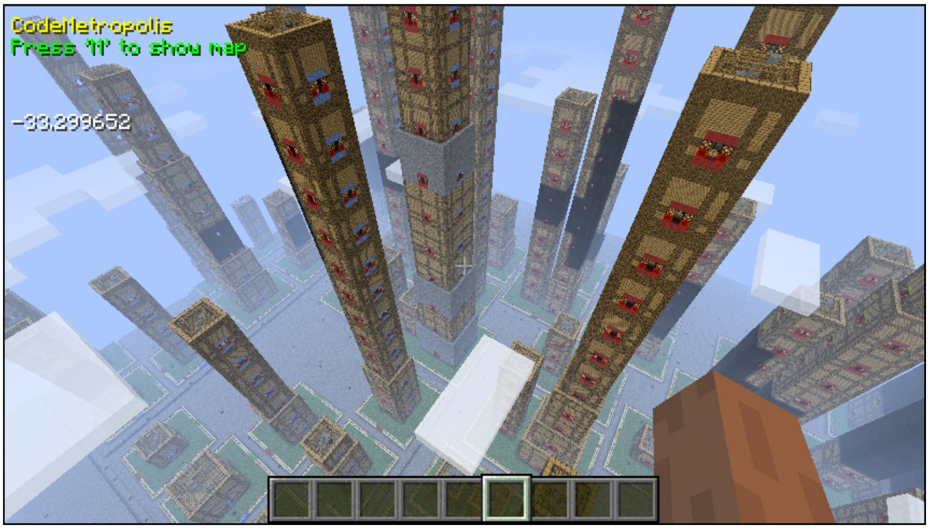


Fig. 1. An typical code metropolis

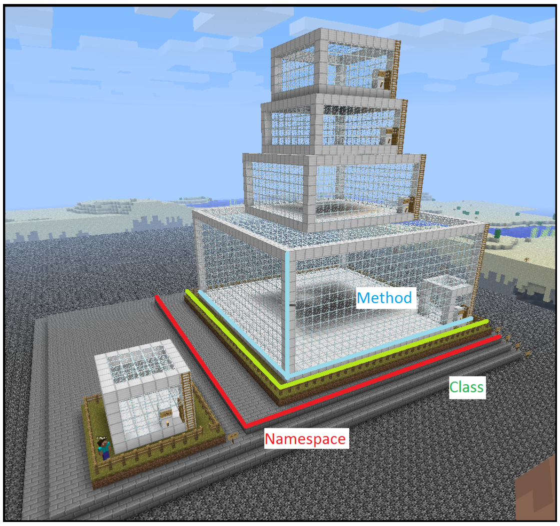


Fig. 2. Items of a city

3 Low-Level Metrics of Virtual Cities

In order to define low-level metrics we have to specify the exact model of a generated city. In this study we mainly focus on cities which only contain buildings like skyscrapers. These buildings could be represented by their bounding box, which is a box with the same width, length and height as the maximal width, length and height of the building itself. The current model does not represent the inner structure of the buildings. The buildings are grouped together into various types, which could be districts at the metaphor level or namespaces and packages at the data level. Buildings also have a position on the plain. Based on the above statements, the current model defines a building as a box with the following properties:

Width the maximal size of the building along the x-axis.
Length the maximal size of the building along the y-axis.
Height the maximal size of the building along the z-axis.
Position an ordered pair of numbers that represents the location of the pivot point of a building on the plain.
Type the unique identifier of the set that the building belongs to.

A city or metropolis is a set of the type of buildings defined here. The buildings cannot be rotated or have any intersecting region.

Three low-level metrics were constructed during the study. Our main design goal was to provide scalable, parameterizable and normalized values which can describe meaningful properties of a city. This means that these metrics have to be independent of the following:

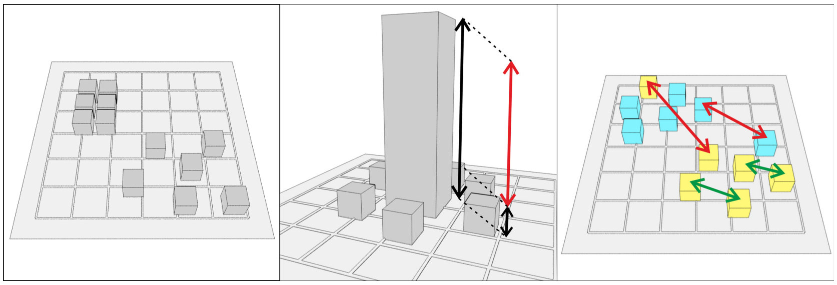
- the area of the city
- the height of the city, i.e. the height of the tallest building
- the number of buildings in the city
- the size of the buildings

With the help of these metrics we can compare the generated metropolises against each other in a formal and automatic way. For example, a huge city like Las Vegas could be compared with a smaller city like Cambridge. Parametrizability is the ability to change the distribution of the metrics, which allows us to fine-tune the values between the lowest and highest possible values, namely zero and one. With these in mind the following metrics were constructed:

Compactness. This expresses the density of the buildings in the city. It is the ratio of the total area of the buildings over the convex hull of the city (Figure 3a).

Homogeneity. This expresses the smoothness of the small scale landscape. It is the distance between any two buildings weighted by the difference in their height. A gamma-correction is used in both parts (Figure 3b).

Connectivity. This describes the coherence among buildings. It is the distance between any two buildings guarded by the their type. A gamma-correction is used in the last parts (Figure 3c).



(a) Compactness (b) Homogeneity (c) Connectivity

Fig. 3. Low level metrics

3.1 Formal Definition

Let us define the buildings as a tuple with 6 items and the collection of these as an unordered set.

$$\mathbb{B} = \{\text{buildings}\} \tag{1}$$

$$b \in \mathbb{B} \tag{2}$$

$$b = \begin{pmatrix} x_b & y_b & z_b \\ |x|_b & |y|_b & |z|_b \end{pmatrix}, \tag{3}$$

where

$$(x_b, y_b, z_b) \in \mathbb{N}^3 \text{ is a predefined pivot point of the building} \tag{4}$$

$$|x|_b, |y|_b, |z|_b \in \mathbb{N} \text{ is the width, the length and the height of the building} \tag{5}$$

$$|\hat{z}|_b = \frac{|z|_b}{\max\limits_{d \in \mathbb{B}} |z|_d} \text{ is the normalized height of the building} \tag{6}$$

We will define the distance between any two buildings as the Euclidean distance between their pivot points.

$$b, d \in \mathbb{B} \tag{7}$$

$$\|b; d\| = \|(x_b; y_b); (x_d; y_d)\| \in \mathbb{R} \tag{8}$$

Furthermore, the following sets will be the corner points of a building:

$$b \in \mathbb{B} \tag{9}$$

$$P_b \subset \mathbb{N}^3 \text{ is the corner points of building } b \tag{10}$$

$$\underline{P}_b \subset P_b \text{ is the lower corner points of building } b \tag{11}$$

$$\overline{P}_b \subset P_b \text{ is the upper corner points of building } b \tag{12}$$

We will also use the convex hull of a building and a set of points:

$$R \subset \mathbb{N}^3 \tag{13}$$

$$\text{Conv}\, R \text{ is the convex hull of point set } R \tag{14}$$

$$D \subseteq \mathbb{B} \tag{15}$$

$$\text{Conv}\, D = \text{Conv}\left(\bigcup_{d \in D} \underline{P}_d\right) \tag{16}$$

The following notation will be used to denote some basic properties of the buildings.

$$b \in \mathbb{B} \tag{17}$$

$$R \subset \mathbb{N}^3 \tag{18}$$

$$A_b = |x|_b \cdot |y|_b \text{ is the area of the building} \tag{19}$$

$$A_R \in \mathbb{N} \text{ is the area of the convex hull of the point set } R \tag{20}$$

$$P_R \in \mathbb{N} \text{ is the perimeter of the convex hull of the point set } R \tag{21}$$

$$D_R = \max_{a,b \in R} \|a; b\| \tag{22}$$

We define a classification over the set of the buildings, this is the type of the building. The type of a building is given with the following relation. It is equal to 1 if and only if two buildings are of the same type:

$$b, d \in \mathbb{B} \tag{23}$$

$$t(b), t(d) \in \mathbb{N} \text{ the type of the building} \tag{24}$$

$$\delta(b; d) \in \{0; 1\} \tag{25}$$

$$\delta(b; d) = \delta(d; b) \tag{26}$$

$$\delta(b; d) = \begin{cases} 1 & \text{if } t(b) = t(d) \\ 0 & \text{if } else \end{cases} \tag{27}$$

Compactness. With these notations, the previously introduced compactness metric could be defined as the ratio of the area of the convex hull of a set of buildings (i.e. the convex hull of the set of points of the buildings) over the total area of these buildings. Because our model does not allow any intersecting buildings, the lower limit will be 0 and the upper limit will be 1.

$$C \subseteq \mathbb{B} \tag{28}$$

$$\text{Comp}\, C = \frac{\sum_{d \in C} A_d}{A_{\text{Conv}\, C}} \tag{29}$$

Connectivity. The third metric called connectivity is defined as the sum of normalized distances between each pair of buildings. We introduced a connection guard and a final normalization part. With this formula we only charge a fee for the buildings with the same type. The size of the fee is greater if the buildings are farther away from each other. This metric was used during a competition to rank the solutions, so gamma correction was applied to the distance part, to be able to fine tune the order.

$$D \subseteq \mathbb{B} \tag{30}$$

$$\underset{\gamma}{\operatorname{Conn}} D = \frac{1}{\binom{|D|}{2}} \sum_{\substack{d,b \in D \\ d \neq b}} \left(\underbrace{\delta(d;b)}_{\text{connection guard}} \underbrace{\left(\frac{\|d;b\|}{\max\limits_{e,f \in D} \|e;f\|} \right)^{\gamma}}_{\text{distance part}} \right) \tag{31}$$

Homogeneity. Next, homogeneity is specified as the arithmetic mean of buildings weighted with the difference of their normalized height. As in the case of connectivity, an overall normalization is added to ensure that the values are bounded. This metric was used during a competition to rank the solutions, so gamma correction was applied to the height difference and to the distance part as well, to be able to fine tune the order.

$$D \subseteq \mathbb{B} \tag{32}$$

$$\underset{\gamma,\zeta}{\operatorname{Hom}} D = \frac{1}{\binom{|D|}{2}} \sum_{\substack{d,b \in D \\ d \neq b}} \left(\underbrace{\left| |\hat{z}|_d - |\hat{z}|_b \right|^{\zeta}}_{\text{height difference part}} \underbrace{\left(\frac{\|d;b\|}{\max\limits_{e,f \in D} \|e;f\|} \right)^{\gamma}}_{\text{distance part}} \right) \tag{33}$$

4 Validation by a User Survey

To validate the previously defined low-level metrics and to create a new high-level metric which is able to express the similarity between a generated city and a real one, a user survey was used. This contained several questions with a predefined list of choices. We asked the users to rank the cities according to their degree of realism. Furthermore, they had to decide which one of the two given cities could be used as an example from a specific point of view. For example, they had to choose the more compact or round one. These examples represented the upper limits of the low-level metrics. To reduce the unintentional influence of the researchers, we used the data of the SEDCup contest, held in the spring of 2013. To win, the teams had to design and implement an algorithm to create an artificial layout of houses with various properties. Only the generated cities were used as trial data in the survey without any preselection. The target audience contained mainly students and coworkers from the IT field. Altogether, 51 complete and 20 partial surveys were processed.

4.1 Validation of Construct of Low-Level Metrics

To address the first research question, the answers from the users were compared with the values of low-level metrics. Because their distributions are unknown, only their directions were considered. It meant that we counted how many times they agreed with the user opinion in the sense that one city had a bigger value than the other. During the evaluation we defined three ranges. If the users gave a positive answer of 40% or less we said that the users disagreed with the metric. If it was more than 40% but less than 60%, it meant that the results were not significant. Lastly, if it was 60% or more it meant that the users agreed with the metric. To avoid any further bias, the users did not know the exact values of the metrics when we did the survey. We summarize our findings on the low-level metrics in Table 1.

Table 1. Agreement among the users and metrics we applied

metric	compactness	connectivity	homogeneity
agreement level	58.62%	78.57%	74.07%
conclusion	not significant	agree	agree

Based on these results, we can answer the first research question presented earlier. It is possible to define low-level metrics for the generated cities that accord with everyday notions, hence appear meaningful. To demonstrate this, we proposed a set of these metrics from which two provide a positive answer. We assume that the meaning of compactness could be better defined so as to improve the level of agreement.

5 Construction of a High-Level Metric

To answer the second research question in the study, a high-level metric was defined. It is able to describe the similarity between a real and a generated city. This metric is the weighted sum of the above-defined compactness, connectivity and homogeneity metrics. We used the answers for the ranking questions of the survey mentioned above. Users had to rank several predefined cities, from the most realistic to the least realistic. The rankings were compared and processed by various methods and algorithms to determine the weights of the low-level metrics.

5.1 Methodology

We used the following methodology to determine the weights for the novel high-level metric called DoR. First of all, the rankings made by users were compared with each other and their correlation was encoded in a graph. The Kendall tau correlation was used to measure the correlation between each pair of rankings. Any pair of observations are said to be concordant if the ranks for both elements agree. With this definition, the degree of correlation is the ratio of the difference between the number of concordant and discordant pairs over the total. That is,

$$\tau = \frac{(\text{number of concordant pairs}) - (\text{number of discordant pairs})}{\frac{1}{2}n(n-1)} \tag{34}$$

The rankings were used as the nodes. The edges connected any two nodes if and only if the Kendall tau correlation coefficient was higher than 0.8.

Next, a community detection algorithm was applied to find the biggest community in the graph. This represents the majority opinion of the users. We used the ranking with the highest number of edges of this community, hence it mirrors the opinions of most of the users. A graph of this is shown in Figure 4. Here, we chose node 48 in cluster 9.

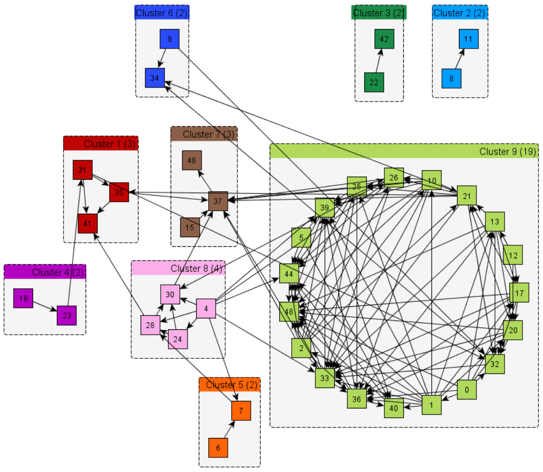


Fig. 4. Correlation graph of the ranking of generated cities

Based of this ranking and the values of the trial data, an inequality system was constructed. It has three free variables that correspond to the weights of the three low-level metrics. Any solution that satisfies these inequalities represents a weight distribution that respects the original ranking of the cities. To find a solution, we define a linear programming problem with a constant target function, hence the simplex method will stop at the first solution. The inequality system does not have any feasible solution, so we had to relax it by removing one of the constraints. This new linear programming problem yielded a solution that respects the original ranking with the exception of one switch. We could not find any better solution. Lastly, we normalized the weights.

5.2 Formal Definition

The trial data used was the city data given in Table 2. We listed the value of the low-level metrics for each with a unique identifier called name. The table also shows the order of cities according to our ranking.

Table 2. Trial data of generated cities along with the values of the low-level metrics

name	**compactness**	**connectivity**	**homogeneity**
Epsilon	0.460	0.969	0.871
Eta	0.123	0.819	0.733
Gamma	0.975	0.611	0.837
Delta	0.582	0.942	0.859
Beta	0.370	0.984	0.873
Alpha	0.799	0.987	0.889
Zeta	0.359	0.999	0.916

This data can be reexpressed as an inequality system, as we did below in matrix form. Using a trivial transformation, it can presented as a linear programing problem. To do this, we have to define the following matrices:

$$L = \begin{pmatrix} 0.46 & 0.969 & 0.871 \\ 0.123 & 0.819 & 0.733 \\ 0.975 & 0.611 & 0.837 \\ 0.582 & 0.942 & 0.859 \\ 0.37 & 0.984 & 0.873 \\ 0.799 & 0.987 & 0.889 \end{pmatrix} \tag{35}$$

$$R = \begin{pmatrix} 0.123 & 0.819 & 0.733 \\ 0.975 & 0.611 & 0.837 \\ 0.582 & 0.942 & 0.859 \\ 0.370 & 0.984 & 0.873 \\ 0.799 & 0.987 & 0.889 \\ 0.359 & 0.999 & 0.916 \end{pmatrix} \tag{36}$$

$$w = \begin{pmatrix} w_{\text{comp}} \\ w_{\text{ecc}} \\ w_{\text{conn}} \\ w_{\text{homo}} \end{pmatrix} \tag{37}$$

$$Lw < Rw \tag{38}$$

$$(L - R)w < 0 \tag{39}$$

$$A = (L - R) = \begin{pmatrix} 0.337 & 0.15 & 0.138 \\ -0.852 & 0.208 & -0.104 \\ 0.393 & -0.331 & -0.022 \\ 0.212 & -0.042 & -0.014 \\ -0.429 & -0.003 & -0.016 \\ 0.44 & -0.012 & -0.027 \end{pmatrix} \tag{40}$$

Because this problem does not have any feasible solution, we will relax it by removing the first inequality from the system. After solving it, a relaxed version with a simplified simplex algorithm, the calculated weights yield a set with an order shown in Table 3.

Table 3. Order of the cities according to the DoR metric

	name	**the** DoR **metric**
→	Epsilon	0.6094
	Eta	0.5137
	Gamma	0.5909
	Delta	0.6009
	Beta	0.6109
	Alfa	0.6209
	Zeta	0.6418

As a final step, we could define our new DoR metric using the following weights:

$$\underset{\zeta,\gamma_1,\gamma_2}{\mathrm{DoR}}\, B = -0.27\, \mathrm{Comp}\, B + -0.73\, \underset{\gamma_1}{\mathrm{Conn}}\, B + 4.43\, \underset{\gamma_2,\zeta}{\mathrm{Hom}}\, B \tag{41}$$

As seen above, we are able to construct a sub-optimal, high-level metric; hence we can give a positive answer to our second research question. In the future, other methods and definition could be investigated and compared with the DoR metric.

6 Conclusions

We live in the age of information explosion where to grasp large amounts of data as quickly as possible is a basic requirement. One of the many possibilities is to convert the data into some clear graphical form, such as data that represents elements of a virtual city. In this study, we presented three computable metrics which express various features of such a city. These are compactness for measuring space consumption, connectivity for showing the low-level coherence among the buildings, and homogeneity for expressing the smoothness of the landscape. These metrics were defined in both a formal and informal way. We also constructed a high-level metric called DoR that is able to express the similarity between a generated metropolis and a real one. Both high- and low-level metrics were validated by a user survey. The opinions obtained in the survey were much as we had anticipated. The results show that it is possible to construct methods (using the DoR metric) which are able to estimate the degree of realism of a generated city. This method embodied as a software-systems could provide a full- or semi-automatic way for creating a life-like virtual environment within a reasonable time. In such a world we could use our everyday senses to perceive the data represented in a clear graphical way.

Acknowledgments. I would like to thank to András London, Norbert Hantos, Tamás Grósz for their help and suggestions during this study and for the evaluation of the results.

References

1. FrontEndART Ltd.: SourceMeter homepage (2014)
2. Lalanne, D., Kohlas, J.: Human Machine Interaction: Research Results of the MMI Program
3. Marcus, A., Comorski, D., Sergeyev, A.: Supporting the evolution of a software visualization tool through usability studies. In: Proceedings of the 13th International Workshop on Program Comprehension, pp. 307–316, May 2005. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1421046 http://dl.acm.org/citation.cfm?id=1058432.1059368
4. Storey, M.A., Best, C., Michaud, J.: SHriMP views: an interactive environment for information visualization and navigation. In: CHI 2002 Extended Abstracts on Human Factors in Computing Systems - CHI 2002, p. 520, April 2002. http://dl.acm.org/citation.cfm?id=506459 http://dl.acm.org/citation.cfm?id=506443.506459
5. Wettel, R., Lanza, M.: CodeCity. In: Companion of the 13th International Conference on Software Engineering - ICSE Companion 2008, p. 921. ACM Press, New York, May 2008. http://scholar.godogle.com/scholar?hl=en&btnG=Search&q=intitle:codecity#1 http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:CodeCity#1 http://dl.acm.org/citation.cfm?id=1370175.1370188
6. Wong, K.: Rigi user's manual. Department of Computer Science, University of Victoria (1998).http://www.rigi.cs.uvic.ca/downloads/pdf/rigi-5_4_4-manual.pdf