CuboidMatrix: Exploring Dynamic Structural Connections in Software Components using Space-Time Cube

Teseo Schneider
Università della Svizzera italiana, Switzerland

Yuriy Tymchuk
University of Bern, Switzerland

Ronie Salgado
Pleiad Lab, DCC, University of Chile

Alexandre Bergel
Pleiad Lab, DCC, University of Chile

***Abstract*—Static and dynamic evolution of software systems may be described in terms of connection additions and removals in a graph. Due to the inherent complexity of software, navigating through such a dynamic network is a non-trivial task and extracting relevant information typically involves sophisticated queries.**

We explore the notion of *space-time cube*, a well-known 3D representation of an evolving dynamic graph, to support a set of software engineering activities. *CuboidMatrix* is a visualization tool that offers simple and expressive navigation operations. We have evaluated our tool against two software comprehension activities, namely (i) assessing interaction of classes during a software execution and (ii) exploring the cause of breaking Lint-like quality rules over a large number of software revisions.

Video companion: https://youtu.be/nOl788-zGE8

Artifact: http://dx.doi.org/10.5281/zenodo.56469

I. Introduction

Software is often made of dynamic networks, typically describing graphs using connected structural elements. For example an object-oriented application is made of objects whose connections change during the application execution. Assessing the evolution of dynamic graphs is a common problem in the software engineering community [1] and visualization community [2].

Space-time cube is a classical visualization in 3D where each data point is visualized in a three-dimensional space, where the depth (*i.e.,* Z-Axis) indicates time. The color and size of each data point indicates metrics and properties of the represented element. Space-time cubes are well known within the visualization community. For example, they have been successfully used to analyze large astronomical data sets [3], and vegetation bush fire [4], to name a few.

The software engineering community has considered using 3D visualization to assist engineering activities [5], [6], however, space-time cube has received scant attention from practitioners to explore software-related data.

CuboidMatrix. In this paper we present CuboidMatrix, a tool using the space-time cube metaphor to represent data. CuboidMatrix offers a rich set of interactions to facilitate navigation and exploration of data sets. In particular, it supports (i) fine control of the camera using an orbital camera movement, (ii) flexible ways to select slices along each of the three axes, (iii) the ability to select a particular data point and its neighbors.

We used CuboidMatrix to support two software engineering activities, namely evaluating the interaction between classes during a program execution and assessing the evolution of Lint-like rules over a large number of software revisions.

Contributions. The paper makes the following contributions:

- Presentation of CuboidMatrix and its set of navigation and interaction facilities.
- Evaluation of CuboidMatrix, our implementation of Space-Time Cube, to solve a software comprehension task.
- Case study using source code quality rules over a large system over a long period of time.

Paper outline. Section II briefly describes the notion of space-time cube and details CuboidMatrix. Section III presents a controlled experiment we have set up to assess CuboidMatrix to solve a software comprehension task. Section IV describes our second experiment on assessing Lint-like quality rules. Section V presents the work related to our effort. Section VI concludes and presents our future work.

II. Space-Time Cube and CuboidMatrix

This section briefly describes the space-time cube visual representation (Section II-A) and details CuboidMatrix, our visualization tool (Section II-B). CuboidMatrix is implemente in the Pharo language[1].

A. Space-time cube

Dynamic networks are networks that change topology and/or edge weights over time. A space-time cube is a visual representation that structures matrices along time, where time is shown as depth. Each slice along the Z-Axis represents a snapshot of the represented network at a given time. A space-time cube is obtained by stacking adjacency matrix representations of the network at each time step. Data points may be colored and be given a weight (represented as its size) that represents a property of the data point. The resulting overall structure is illustrated in Figure 1.

[1] http://pharo.org/

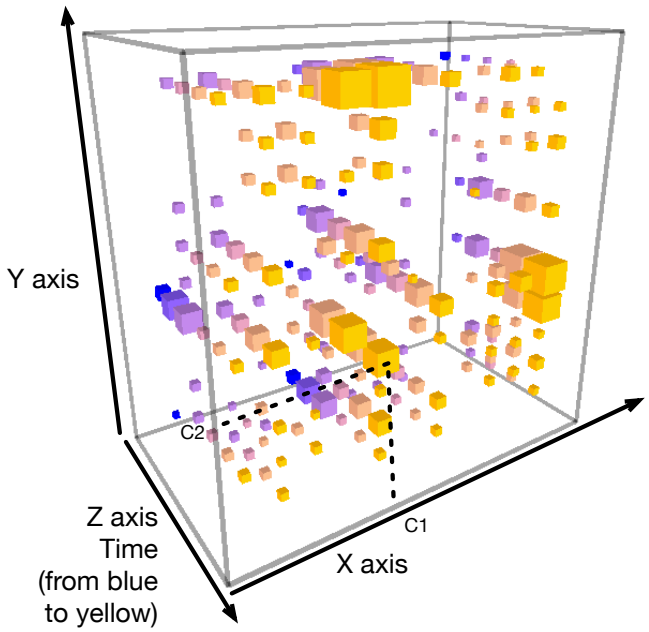

Figure 1: Space-time cube principle

The figure is a screenshot of our implementation of space-time cube. As an illustration, the figure uses a dataset representing interaction between classes during a program execution. Each data point, represented as a cube, indicates an interaction between two classes. Each time unit (mapped on the Z-Axis) represents a wall-clock time period (a couple of milliseconds). The size of a cube reflects the amount of interaction between the two designed classes at a particular time frame. We represent time using a color scaling from blue to yellow. Blue indicates the past and yellow the future. Passing time is indicated by a blue-to-yellow fading.

Formally, each cube, denoted $(C1, C2, T)$, represents a relation between two elements $C1$ and $C2$ at a given time slice T. The size of the cube indicates a metric value for that relation, and the color is a visual support to indicate T.

B. CuboidMatrix

CuboidMatrix uses a space-time cube to render the set of data, as previously described. To ease the exploration of data, CuboidMatrix offers a number of ways to interact with the data set.

Camera movement. The visible portion of the cube and the view orientation are given by a camera. The camera can be moved by using the mouse in an orbital fashion. The camera movement is perceived as a whole rotation of the cube while keeping the point of view at the center of the cube. The orbit follows a trajectory on a sphere.

The radius of the sphere can be increased or decreased, using the keys `W` and `S`. This effect is perceived as zooming in and out, respectively.

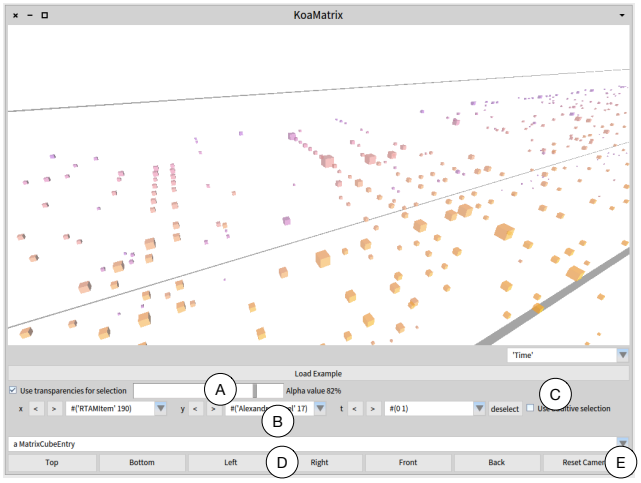

Figure 2: The CuboidMatrix tool

Initially, the camera is oriented toward the center of the space-time cube. The point of view may be modified with the keys `A` and `D`, to move it to the left and to the right, respectively. The keys `Q` and `E` move the point of view up and down, respectively. The camera may be reset to its original position (button `E` in Figure 2).

Tooltip and data point selection. Data about an individual data point may be obtained by simply locating the mouse above it. The exact data point is provided as a tool tip, including its weight, and the three coordinates.

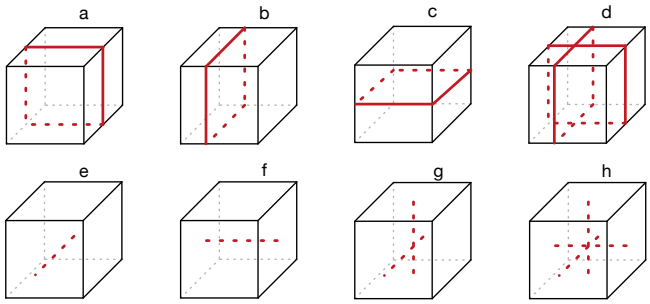

Figure 3: Selection strategies

Clicking on a data point triggers a selection. The selection highlights some data points that are deduced from a strategy. We provide 8 selection strategies, as depicted in Figure 3. Clicking on a data point selects all the data points in (a) the same time frame; (b) in the same vertical slice, along the Y-Axis; (c) along the X-Axis; (d) using a combination of strategies a and b; (d) in the same X-Axis or Y-Axis; (e) data points in the same X and Y axes; (f) in the same Y and Z axes; (g) in the same Z or Y axes; (h) select all the data points sharing two coordinates. A selection is indicated with a particular color palette and selected data points are completely opaque.

Slice selection. A control panel is dedicated to selecting particular slices (B in Figure 2). The control panel supports the selection of a particular slice along each of the 3 axes. Two buttons, < and >, allow for movement of the slice along an axis.

A drop down menu allows for selecting a slice. This feature is useful in the presence of a dense space-time cube where the pointing using the mouse is not effective.

Additive selection. By default, selecting a data point deselects the previously selected points. A checkbox, marked C in Figure 2, allows for maintaining previously selected points in place when selecting a new point.

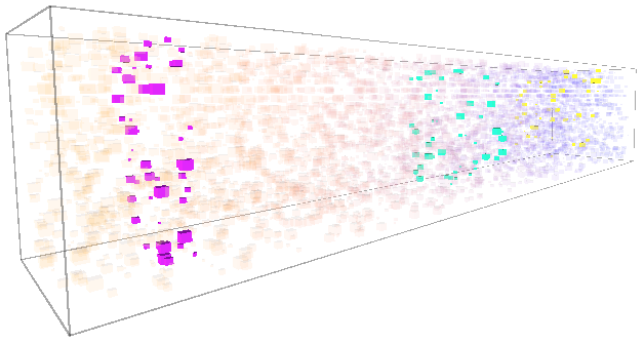

Figure 4: Additive selection of multiple time slices

Figure 4 illustrates additive selection by having three selected time slices. This feature helps to ccompare directly different sets of data points in the visualization.

Alpha channel. Some visual elements may hide other elements. This effect is called occlusion and is a serious concern in a 3D visualization. The common way to address this problem is to use transparency [6]: the hiding effect of a data point is reduced when transparent.

CuboidMatrix allows the user to manually set the transparency for data points that are not selected. A slider (A in Figure 2) controls the transparency of data points. Data points may be completely transparent and therefore not part of the view.

Projections. The space-time cube may be projected along each side, as a 2D projection using a control panel marked D in Figure 2. Figure 5 illustrates the projections of a dataset. When projected, the space-time cube is rendered using an orthographic projection from a side, without perspective. A dataset may be projected along each of the six sides. The figure shows four of them.

In contrast with the classical projection where distant objects appear smaller, with the orthographic projection all data points have the size reflected by their weight, independently of the distance between the data point and the camera. For instance, by looking at the space-time cube from the front (Figure 5, `Front`), all the time-slices are flattened and the transparency contributions of each cube are added. In this way the opacity of the cubes illustrates the persistence of the entities through time.

The projections offered by CuboidMatrix are inspired by Cubix [3].

III. Control Experiment: Assessing Class Dependencies

This section presents a controlled experiment to assess the performances of CuboidMatrix against a software comprehension task.

A. Motivation & Datasets

Object-orientation promotes the use of message sends (also called "method call") to model a computation. During a program execution, objects are created and these objects are sending messages to each other. Understanding the exact interaction between objects and classes is known to be a serious concern among practitioners [7].

To assess CuboidMatrix, we will use two software comprehension tasks, based on class interactions during a program execution. Each of the tasks uses a particular dataset. Each dataset is obtained by monitoring the execution of a program. Figure 6 illustrates CuboidMatrix on one of the datasets.

B. Data sets

As we will use two comparable tasks in our experiment, we have produced two datasets obtained from the execution of two different applications:

- D_1 – The first dataset is composed of 621 data points. It shows the interaction of 101 different classes along a time frame of 19 periods. The space-time cube has a dimension of $56 \times 45 \times 19$.
- D_2 – The second dataset is composed of 652 data points. It is divided into 19 time periods and contains interaction between 60 classes. The size of the space-time cube is $34 \times 26 \times 19$.

In both data sets, all the data points have a weight of 1. This means that all the cubes have the same size. D_1 and D_2 are comparable, albeit not identical. D_2 is dense compared to D_1 since it has more data points and the space-time cube is smaller. These datasets have the particularity of having persistence in time, which means that two classes interacting at a time t have a likelihood of interacting at $t+1$.

The two datasets are represented within the CuboidMatrix as follows: a data point (c_1, c_2, t) indicates that during the time slice t, instances of the class c_1 sent messages toward instances of the class c_2. The data sets were obtained using the Spy profiling framework [8].

C. Excel as the baseline

A baseline is necessary to compare CuboidMatrix against. Such a baseline has to be carefully chosen as it represents

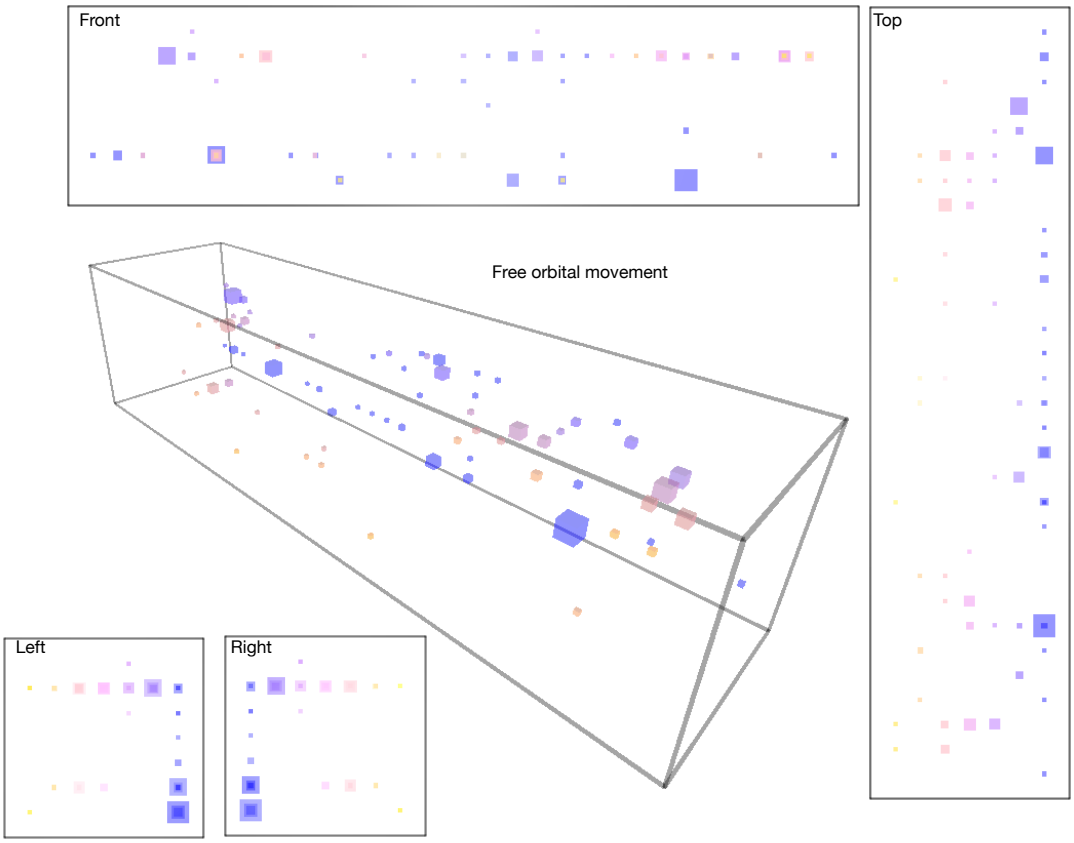

Figure 5: Orthographic projections

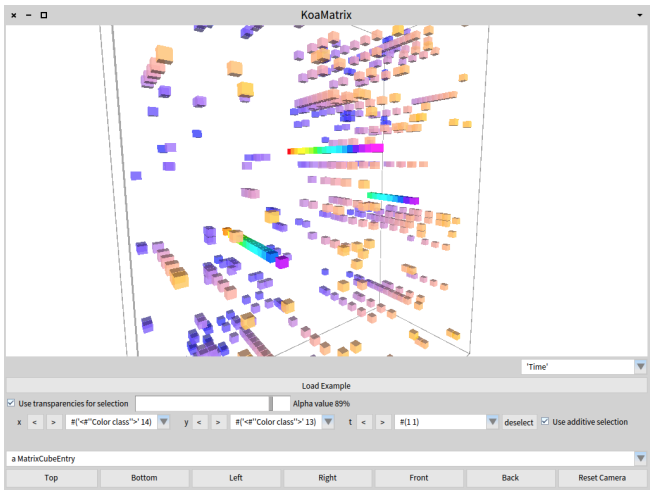

Figure 6: CuboidMatrix with the class dependency dataset

the control of our experiment. Two criteria are relevant when identifying the baseline:

- *Fairness* – The baseline has to be impartially selected without supporting an approach that will naturally favor CuboidMatrix for a particular set of tasks.
- *Representative* – The baseline has to represent current practices to solve situations in which we plan to use CuboidMatrix. Moreover, the baseline has to provide a typical and natural solution for the problem to solve.

Excel. We have opted for the use of *Excel* as the baseline to compare CuboidMatrix. Excel is often regarded as a natural choice to manipulate tabular data. At first, this choice seems unfair: Excel presents numerical values in a possibly long textual list which makes little use of the cognitive ability of a programmer. However, Excel offers a large range of operations to manipulate data: one can filter, order, transform, and create charts. Macros can also be defined by using a large set of predefined functions. Excel also supports macros written in VisualBasic, which is a full generic-purpose programming language. Excel has a complete built-in help support.

Excel also has the advantage of being known by a large population. We therefore expect basic operations in Excel, such as defining simple macros, transforming, filtering, and ordering data to be known by the participants. Solving specific problems using Excel is also largely discussed online. Excel has been used as a baseline in another similar experiment [5].

Discarded baselines. A large range of solutions have been proposed by the software and data visualization community. However, none of the proposed solutions have gained enough attraction to be considered a "standard solution" to solve the tasks we defined (Section III-A). Using a non-standard and/or little known approach may therefore be unfit for our experience and may invalidate our results.

Having our datasets in an SQL database may appear as a reliable alternative baseline. SQL is a rich language to transform tables and compute non-trivial queries. There is a risk however, that many participants are unfamiliar with SQL. We may therefore introduce a significant advantage for CuboidMatrix if non-SQL experts participate in our experiment. Since we wish to run our experiment on what may be perceived as "average software engineers", we opted out of SQL in favor of Excel.

D. Experiment design

Generic work session. The activity of a participant is structured as follows:

1) *Excel learning material*: Participants are expected to have a minimal command of Excel. This learning material provides screenshots that summarize the ordering and filtering features of Excel.
2) *CuboidMatrix learning material*: A concise description of CuboidMatrix is provided. The different interactions are also presented. This material is a condensed version of Section II.
3) *Exercise 1*: Answering the set of questions for a given exploration tool (CuboidMatrix or Excel) and data set (D_1 or D_2).
4) *Exercise 2*: Answering the set of questions for the other exploration tool and the other data set.
5) *Open retrospective*: We ask some open questions regarding the experiment. We also informally and orally get the impression of the participant.

Since we use two datasets and two exercises, we have defined four specific work sessions:

Work	Exercise 1		Exercise 2	
session	Dataset	tool	Dataset	tool
W_1	D_1	EX	D_2	CM
W_2	D_2	CM	D_1	EX
W_3	D_2	EX	D_1	CM
W_4	D_1	CM	D_2	EX

We refer to the use of Excel as EX and the use of CuboidMatrix as CM. Work sessions W_1 / W_2, W_3 / W_4 reverse the order of the exercises. This is useful for minimizing a bias that may be due to a learning effect against a particular configuration.

The measured and tested dependent variable is the productivity of solving an exercise and the independent variable reflects the tool used.

Questionary. Each exercise consists of five questions:

- Q1 - Which classes $c1$ interact with $c2$ only during the three first time periods? We are therefore looking for $c1$ and $c2$ for which we have a relation at $(c1, c2, 1), (c1, c2, 2)$, and $(c1, c2, 3)$.
- Q2 - Which classes interact at the end of the execution, only during the last three time periods?
- Q3 - Which are the two classes that interact during the overall execution?
- Q4 - Which classes $c1$ interact with $c2$ only at time periods that are even (2, 4, 6, ..., 18)?
- Q5 - Can you identify some interactions that are identical, along time, between groups of classes? We are looking for groups of classes $\{(c1, c2), (c3, c4)\}$ for which we have data points $(c1, c2, T_1), ..., (c1, c2, T_n)$ and $(c3, c4, T_1), ..., (c3, c4, T_n)$

We use the exact same set of questions for the two datasets. Questions Q1, Q2, Q3, Q4 are about the interaction between two classes along the execution. Answering Q5 involves identifying groups of classes. We therefore expect this last question to require more effort to answer than the others.

Oracle. We have algorithmically computed the exact answer of each of the questions for both data sets to be able to quantitatively evaluate the results. Questions Q1, Q2, Q3, and Q4 have exactly one answer. Question Q5 has 9 groups of interacting classes using dataset D_1 and 27 groups using D_2.

Scoring. A participant is free to provide many answers per question. The exercise description and the question phrasing do not let the participant guess the amount of correct results per question.

For questions Q1, Q2, Q3, and Q4, we will use the following scoring function:

- Score = 1 if the provided result is exactly the one provided by the oracle;
- Score = 0.5 if the provided result contains the exact answer from the oracle, but contains some false-positive;
- Score = 0 otherwise.

The question Q5 requires looking for a different and more complex pattern than for the other questions. We therefore use a different scoring, as follows:

- Score = 1 if all the provided results for Q5 are correct, but not all the correct answers are required;
- Score = 0.5 if at least one of the provided results is correct
- Score = 0 otherwise

We could have used the standard precision and recall metrics to score each answer. However, this would have made the analysis more complex. Giving a score of 0, 0.5, or 1 to each answer simplifies the way each participant output is analyzed.

For each exercise and each participant p, a score $S(p) = \sum_{n=1}^{5} \mathit{Score}(Q_n, p)$ is given, ranging from 0 to 5. A score

Part.	Exercise 1					Exercise 2					EX	CM
	Q1	Q2	Q3	Q4	Q5	Q1	Q2	Q3	Q4	Q5		
P1	0	0	0	0	0	1	1	1	1	1	0	5
P4	1	1	1	1	0.5	1	1	1	1	1	4.5	5
P6	0	0	0	0	0	0.5	0.5	0	0	0	0	1
P8	0	0	0	0	0	1	1	1	1	1	0	5
P2	0.5	0.5	1	1	1	0.5	0.5	1	1	0	3	4
P3	1	1	1	0	1	0	0	0	0	0	0	4
P5	1	1	1	1	1	1	0	0	0	0	1	5
P7	0.5	0	1	0	0	0.5	0.5	0	0	0	1	1.5

Figure 7: Raw result of the controlled experiment (Excel result in gray, EX refers to Excel and CM to CuboidMatrix)

of 5 indicates that the participant has correctly answered all the questions. A score of 0 means that none of the answers is correct.

E. Pilot study

Prior to running the full experiment, we ran a pilot study on two participants. This pilot study enabled us to fine tune the experiment. In particular, the pilot helped us on the following points:

- The pilot showed that the two exercises can be solved by both Excel and CuboidMatrix.
- The two participants have answered each exercise question in slightly less than 30 minutes. We therefore set 30 minutes as the maximum allowed time to complete an exercise.
- One of the participants felt the need to look for documentation online. We therefore let the participants to seek online for help for the full experiment.
- Our original presentation of CuboidMatrix, given in the learning material, was longer. We therefore shortened it to help the participant to fully read it and understand it.
- In our original formulation the questions contained some ambiguities and were not clear. We therefore have improved the question formulation.

F. Results

Participant profile. We ran our experiment on 8 participants. Among these participants, there were 3 professional software engineers, 2 PhD students, 2 master students, and 1 undergrad student.

Raw data. Figure 7 details the answers and overall results for each participant. The first column gives the participant identifier. The following 10 columns give the scores for each question of Exercise 1, followed by those of Exercise 2. The last two columns sum the individual scores for Excel and for CuboidMatrix. Values referring to the Excel scoring is marked in gray to ease the reading.

Analysis. According to the number of participants, we use the Mann-Whitney test (two-tailed, with a confidence interval of 95%) to analyze the last two columns of Figure 7. This test is a nonparametric test useful when no assumption can be made on the normality of the data (which is our case). We apply this test to measure the effect of the tool on the participant score.

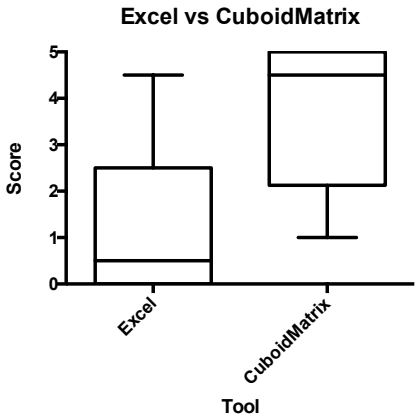

Figure 8: Result analysis

Figure 8 gives the distribution of the scores for the two tools we are considering. The boxplot indicates that the score for Excel has a mean of 1.1 and a median of 0.5. The standard deviation is 1.6. The score of CuboidMatrix has a mean of 3.8, a median of 4.5, and a standard deviation of 1.6.

The Mann-Whitney test on the exercise scores indicate that the scores obtained for Excel and CuboidMatrix significantly differ (with a p value of 0.0057), in favor of our tool.

Observing the use of Excel. We observed the participants while they perform the experiment. We report here the most notable development aspects of the experiment.

All but one participant used macros: four participants defined macros using the Excel functions COUNTIF and SUMIF, two participants defined the macros in VisualBasic. An interesting aspect of using macros is that nearly all the participants spent a significant portion of the allowed 30 minutes searching for documentation online. Almost no one used the inline Excel help. Participant P4 did not use macros and obtained the highest score. Instead, the participant answered the questions only by filtering and ordering columns. P4 is experienced retrieving data using Excel functionalities. He also did not use macros since he feels uncomfortable using them.

Participant P2 transformed two columns into an adjacency matrix (Excel supports this transformation using a dynamic table). However, such an adjacency matrix did not produce a useful result. Instead, P2 had to fall back on using macros and manually searching.

All of the participants, except P4, looked for information online. Here are some of the online queries: “How to find and select duplicate rows in a range in Excel”, “How to get distinct or unique values in a column in Excel”, “Help in defining VisualBasic macros to concatenate columns of data

in Excel", "Accessing worksheet cells from VisualBasic", "How to loop through a list of data on a worksheet by using macros in Excel".

Observing the use of CuboidMatrix. Overall, the usages of CuboidMatrix were generally in line with what we expected: participants have used the features described in the learning material. Questions Q1 and Q2 were answered using the additive slice selection. Question Q3 required the additive selections with a low alpha color value for non-selected time lines. Question Q4 was answered using the additive slice selection: participants selected time slices 2, 4, 6, ... 18, to get the relevant result.

Q5 was probably the most complex question. Most of the participants were unstructured when trying to answer it (note that being unstructured does not imply providing a wrong answer). Interestingly, P4 used slice selections along the Y-Axis to answer P5.

Open retrospective. After the experiment, we questioned the participants about the exercises. Their answers were not graded in front of the participants to make sure that each participant was not pressured to answer.

When using CuboidMatrix, P1 strongly felt that the occlusion was a problem, particularly when analyzing data points located at the center of the space-time cube. Nevertheless, this perceived problem did not prevent the participant from getting the best score.

No participant felt that the CuboidMatrix or Excel were inadequate to answer the questions. Also, the set of questions were perceived as representative tasks a software engineer may have to face.

All the participants expressed a positive feeling toward CuboidMatrix. P5 was the most enthusiastic about it. The participant found the visualization very intuitive. Interestingly, the participant happens to be a fan of 3D games.

Conclusion. The controlled experiment measures the ability of the participants to solve two comparable exercises using two different tools: Excel and CuboidMatrix.

We took great care to be fair when designing our experiment. We provided all the necessary help participants requested regarding the questions and the tools. Participants were also free to look for information online.

Our experiment shows that participants performed significantly better using CuboidMatrix than using Excel. We therefore conclude that for the provided set of datasets and questions, CuboidMatrix significantly outperforms Excel.

IV. Case Study: Quality Rules

As a second evaluation, we conducted a case study on assessing the evolution of Lint-like code quality rules over several versions of a large software system.

A. Motivation

Lint is a utility that flags a source code portion according to a set of source code pattern matching rules. Each rule looks for source code anomalies. These rules, when applied on a software system, produce *critics*. A critic indicates a rule violation on a particular piece of code.

In the Pharo ecosystem [9], SmallLint and QualityAssistant [10] are used to catch defects and possible bug introduction early on at each release of Pharo. One essential question the Pharo community has is whether the overall source code quality has evolved since the introduction of these two tools.

In particular for this case study, we considered the following questions:

1) Which packages have the most variations in critics?
2) Quality based on which critics have changed the most: Are there any rules with a constant ascending or descending critics production? Are there any rules variation of the amount of produced critics?
3) Are there any rules with similar patterns of critic production variation?

The original data set consists of critics generated by 120 rules about 240 packages for 680 versions.

B. Analysis & Conclusion

First of all, the dataset to analyze is large, which produce a visualization composed of tiny elements. The size of the elements is so small that we are not able to assess the difference between them. Reducing the size of the matrix was therefore an important step to comfortably conduct our analysis.

We extracted a subset composed of 10 rules (we selected the one with an "error" severity level), 20 packages (we selected the packages that are the most frequently modified), and 100 versions that contained the greatest variations in the generated critics. As CuboidMatrix does not support strategies to compact a dataset, we had to manually extract this subset. The obtained visualization is shown in Figure 9.

Although we greatly reduced our dataset, it remained challenging to detect the differences between data points since the produced visual elements looks similar. The maximum number of critics is 40 and most of the changes between adjacent data points are equal to 1 or 2 critics. Since CuboidMatrix uses a linear interpolation for the data point weight, variations of 1 or 2 units are not easily perceptible. The orthographic projections helped us understand the amount of critic variations since comparisons are easier to carry out without a perspective.

Figure 10 illustrates the bottom projection where the Y-Axis represents the packages while the X-Axis represents time. While we cannot identify any trends, we can determine which packages have the most critics. Translucent overlay allows us to identify not only the biggest number of critics from a single rule, but also the other ones as concentric data points (*i.e.,* data points sharing the same center) of a darker shade. However it is harder to detect this kind of data on

the yellow spectrum of the visualization as it tends to blend more.

Figure 11 represents the frontal projection, where the Y-Axis represents the quality rules and the X-Axis represents the packages. This projection highlights which packages are not affected by any rules, and which rules produced critics about most of the packages. This projection also allows us to see which critics have changed over the timespan. These critics can be selected to highlight changes. Figure 12 shows a selected sequence where the data point sizes are decreasing from left to right. However, Figure 13 shows exactly the same view but without selection, and in this case, it is difficult to see the decreasing trend because of the translucency of the cubes and the perspective.

We ran into issues with scalability and were not able to analyze the whole dataset. We had problems with change' detection on the subset that we were trying to analyze, but with the help of projections and selections we were able to partially overcome these issues.

V. Related Work

The related work is structured along two complementary axes: visualization tools for software and data. Representing the evolution of networks has received a significant effort from the research community. For example, the visual survey http://dynamicgraphs.fbeck.com provides a significant list of related work. This section presents the most relevant piece of work related to this paper.

A. Software engineering tools

Software as cities. Software visualized as a city is probably the most popular 3D software visualization.

CodeCity [11] uses a city metaphor where each package is a district, visually represented as a flat rectangular square. On top of a district, buildings are located. Each building represents a class for which the height is the number of methods, and the width is the number of attributes. The color of a building indicates the number of lines of code of the represented class. CodeCity has been evaluated using an experiment similar to the one we have set up. Designed tasks are carried out by some participants using CodeCity and Excel.

EvoStreets [12] uses a three-staged representation chain. The geographic landscape is collected from a primary model. Aspects of the landscape are obtained from a secondary model. Tertiary models define the colors and superposed diagrams. Contrary to CodeCity, EvoStreets combines decomposition hierarchy, element properties and development time.

Verso. Langelier *et al.* [13], [14] proposed Verso, a 3D visualization of evolving systems. In their approach, classes are represented as 3D boxes, arranged over a flat 2D plan. Java interfaces are rendered using a cylinder. A set of graphical characteristics are mapped to metrics: color, height, width, and rotation. A kind of treemap algorithm is used as layout. The camera follows an orbital movement, as in CuboidMatrix. Animation is used to show evolution of the data set.

Software Dynamicity. Verso [13] has been extended with a heatmap. The heatmap represents basic properties related to time or a combination of such properties [15].

sv3D. Marcus *et al.* [16] explored the use of 3D to visualize lines of source code. The sv3D tool [6] uses a box to represent a line of code. The box color indicates its control structure type (*e.g.,* a loop, a conditional statement). The size of the box indicates its nesting level. Deeply nested structures, such as algorithms, appear composed of large boxes.

sv3D authors compared their visualization against the actual source code. The authors have formulated two hypotheses regarding the accuracy and the task completion time and three groups of students were used in a controlled experiment.

B. Data visualizations

The space-time cube metaphor is intensively employed in the field of data visualization. The website `http://spacetimecubevis.com` covers popular uses of that metaphor.

Cubix. Bach *et al.* [3] designed Cubix, a 3D representation of space-time cube, in which slices may be unfolded into adjacency matrices. Cubix inspired CuboidMatrix in many aspects (*e.g.,* orthographic projections). However, CuboidMatrix offers original interactions such as the slice selection and support additional selection that are key to complete the comprehension tasks we have designed.

ExTraVis. Cornelissen *et al.* [17], [18] proposed ExTraVis a sequence of circular bundle views to represent execution traces and allows one to navigate through trace elements.

Animation. Animation is a common technique to represent dynamic networks. In fact, any static network visualization can be used to generate one image per time step. These visualizations are then displayed one after the other in an animation. In this context the main challenge is to highlight and show the differences in an effective coherent way [19], [20], [21], [22], [23], [24].

snapshots. The alternative to animation consist of laying out the different snapshots one after the other [25], [26], [27], [28]. An interesting exception consists of laying the snapshots along the depth, creating a 2.5D visualization [29]. Time dependent cubes may be effectively represented using adjacency matrices. Small MultiPiles [30] is a technique that consists in piling adjacency matrices. Such matrices can be aggregated and a small widget, called flipbook, allows one to select a matrix within the aggregation.

VI. Conclusion and Future Work

Due to the difficulty is finding reliable metaphors, 3D software visualization has had little impact within the software

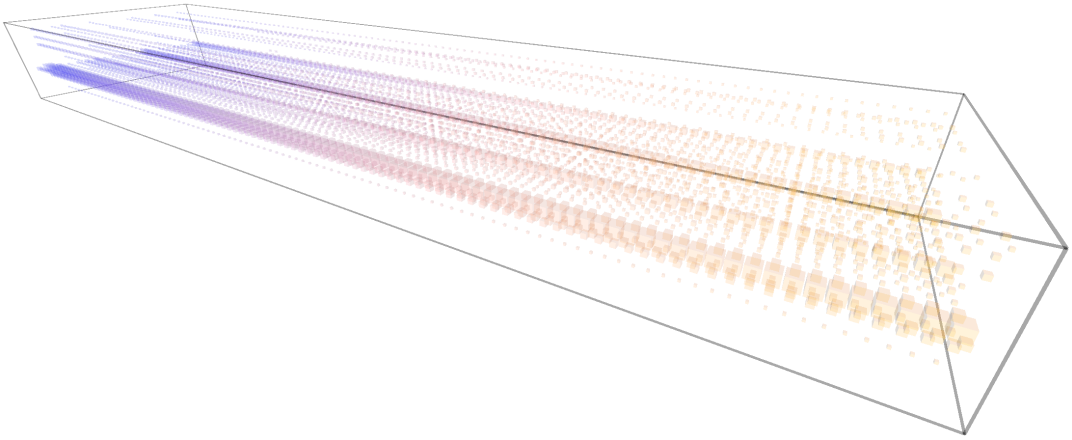

Figure 9: Evolution of SmallLint rules over multiple revisions of Pharo (time goes from blue to yellow)

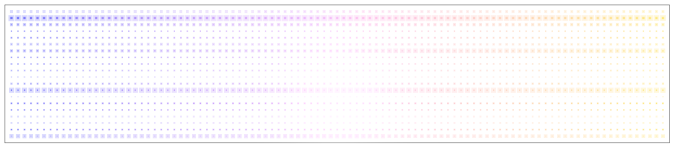

Figure 10: Bottom orthographic projection

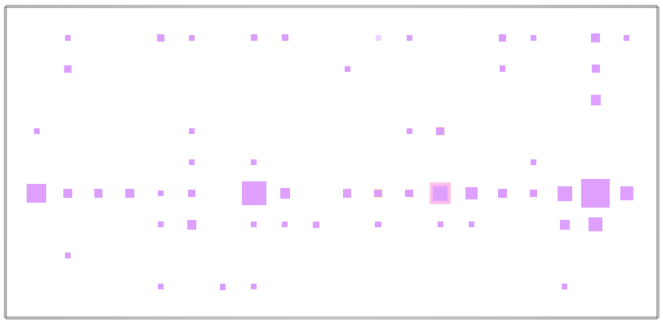

Figure 11: Orthographic projection illustrating the irregularities in considering the rules

Figure 12: Visible changes in a selected sequence

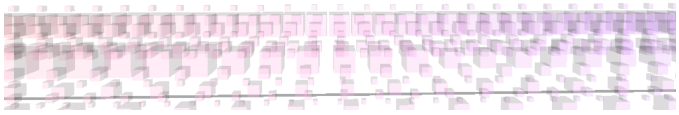

Figure 13: No visible changes when nothing is selected

engineering community. This paper presents CuboidMatrix, a visualization tool that employs the space-time cube metaphor to navigate within a dataset and interact with it. CuboidMatrix was used in a controlled experiment involving two software comprehensions tasks. Our results show that our tool is significantly better than Excel to solve the two exercises we defined related to software execution analysis. We conducted a case study that highlights some of the limitations of our current implementation of CuboidMatrix.

As future work, we plan to address the following two points:

- *Element ordering* – Despite our effort to provide interactive options to avoid occlusion, some participants in our experiment complained about the occlusion. In particular, data points contained in the center of a visualization are difficult to relate from other points. As future work, we will investigate the use of ordering along both axes to ease the reading of data points located at the center of the visualization.
- *2D vs 3D* – The relevance of using a three-dimensional visualization has long been debated [6], [31] in the software visualization community. As far as we are aware, it has not been demonstrated that using three dimensions is a significant improvement over using two dimensions. In the future, we will compare our visualization against 2D adjacency matrices.

Acknowledgment

We thank Renato Cerro for his comments on an early draft of this paper. We also thanks ESUG (www.esug.org) for partially sponsoring this work. This work was supported by the SNF under project number 200020_156178.

References

[1] T. Gîrba and S. Ducasse, "Modeling history to analyze software evolution," *Journal of Software Maintenance: Research and Practice (JSME)*, vol. 18, pp. 207–236, 2006.

[2] X. Liu and H.-W. Shen, "The effects of representation and juxtaposition on graphical perception of matrix visualization," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 2015.

[3] B. Bach, E. Pietriga, and J.-D. Fekete, "Visualizing dynamic networks with matrix cubes," in *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*. 2014.

[4] B. Bach, P. Dragicevic, D. Archambault, C. Hurter, and S. Carpendale, "A Review of Temporal Data Visualizations Based on Space-Time Cube Operations," in *Eurographics Conference on Visualization*, 2014.

[5] R. Wettel, M. Lanza, and R. Robbes, "Software systems as cities: a controlled experiment," in *Proceedings of ICSE '11*.

[6] A. Marcus, L. Feng, and J. I. Maletic, "3d representations for software visualization," in *Proceedings of SoftViz '03*.

[7] J. Sillito, G. C. Murphy, and K. De Volder, "Asking and answering questions during a programming change task," *IEEE Trans. Softw. Eng.*, vol. 34, pp. 434–451, Jul. 2008.

[8] A. Bergel, F. Bañados, R. Robbes, and D. Röthlisberger, "Spy: A flexible code profiling framework," *Journal of Computer Languages, Systems and Structures*, vol. 38, no. 1, Dec. 2011.

[9] A. Bergel, D. Cassou, S. Ducasse, and J. Laval, *Deep Into Pharo*. Square Bracket Associates, 2013.

[10] Y. Tymchuk, "What if clippy would criticize your code?" in *BENEVOL'15: Proceedings of the 14th edition of the Belgian-Netherlands software evoLution seminar*.

[11] R. Wettel and M. Lanza, "Visualizing software systems as cities," in *Proceedings of VISSOFT 2007*.

[12] F. Steinbrückner and C. Lewerentz, "Understanding software evolution with software cities," *Information Visualization*, vol. 12, no. 2, pp. 200–216, Apr. 2013.

[13] G. Langelier, H. A. Sahraoui, and P. Poulin, "Visualization-based analysis of quality for large-scale software systems," in *Proceedings of ASE '05*.

[14] G. Langelier, H. Sahraoui, and P. Poulin, "Exploring the evolution of software quality with animated visualization," in *Proceedings of VLHCC '08*.

[15] O. Benomar, H. Sahraoui, and P. Poulin, "Visualizing software dynamicities with heat maps," in *Proceedings of VISSOFT '13*.

[16] A. Marcus, D. Comorski, and A. Sergeyev, "Supporting the evolution of a software visualization tool through usability studies," in *Proceedings of the 13th International Workshop on Program Comprehension*. 2005.

[17] B. Cornelissen, D. Holten, A. Zaidman, L. Moonen, J. J. van Wijk, and A. van Deursen, "Understanding execution traces using massive sequence and circular bundle views," in *Proceedings of ICPC '07*.

[18] B. Cornelissen, A. Zaidman, D. Holten, L. Moonen, A. van Deursen, and J. J. van Wijk, "Execution trace analysis through massive sequence and circular bundle views," *J. Syst. Softw.*, vol. 81, pp. 2252–2268, Dec. 2008.

[19] C. Erten, S. G. Kobourov, V. Le, and A. Navabi, "Simultaneous graph drawing: Layout algorithms and visualization schemes," in *Proceedings of Graph Drawing '04*.

[20] C. Erten, P. J. Harding, S. G. Kobourov, K. Wampler, and G. V. Yee, "GraphAEL: Graph animations with evolving layouts graph drawing," in *Proceedings of Graph Drawing '04*.

[21] D. Beyer and A. E. Hassan, "Animated visualization of software history using evolution storyboards," in *Proceedings of WCRE '06*.

[22] B. Bach, E. Pietriga, and J.-D. Fekete, "Graphdiaries: Animated transitions and temporal navigation for dynamic networks," *IEEE Trans. Vis. Comput. Graphics*, vol. 20, no. 5, pp. 740–754, May 2014.

[23] P. A. Grabowicz, L. M. Aiello, and F. Menczer, "Fast filtering and animation of large dynamic networks," *EPJ Data Science*, vol. 3, no. 1, pp. 1–16, 2014.

[24] C. Ma, R. V. Kenyon, A. G. Forbes, T. Berger-Wolf, B. J. Slater, and D. A. Llano, "Visualizing Dynamic Brain Networks Using an Animated Dual-Representation," in *Proceedings of EuroVis '15 - Short Papers*.

[25] C. Friedrich and P. Eades, "Graph drawing in motion," *Journal of Graph Algorithms and Applications*, vol. 6, no. 3, pp. 353–370, 2002.

[26] C. Collberg, S. Kobourov, J. Nagra, J. Pitts, and K. Wampler, "A system for graph-based visualization of the evolution of software," in *Proceedings of SoftViz '03*.

[27] A. Perer and J. Sun, "MatrixFlow: Temporal network visual analytics to track symptom evolution during disease progression," in *Proceedings of the AMIA Annual Symposium*. American Medical Informatics Association, 2012, p. 716.

[28] A. B. Alencar, K. Börner, F. V. Paulovich, and M. C. F. de Oliveira, "Time-aware visualization of document collections," in *Proceedings of SAC '12*.

[29] M. Gaertler and D. Wagner, "A hybrid model for drawing dynamic and evolving graphs," in *Proceedings of Graph Drawing '06*.

[30] B. Bach, N. Henry-Riche, T. Dwyer, T. Madhyastha, J.-D. Fekete, and T. Grabowski, "Small multipiles: Piling time to explore temporal patterns in dynamic networks," *Computer Graphics Forum*, vol. 34, no. 3, pp. 31–40, 2015.

[31] A. Kerren and F. Schreiber, "Why integrate infovis and scivis?: An example from systems biology," *IEEE Comput. Graph. Appl.*, vol. 34, no. 6, pp. 69–73, Nov. 2014.