Research Perspective on Supporting Software Engineering via Physical 3D Models

Florian Fittkau, Erik Koppenhagen, and Wilhelm Hasselbring
Software Engineering Group, Kiel University, Kiel, Germany
Email: {ffi, eko, wha}@informatik.uni-kiel.de

***Abstract*—Building architects, but also civil or mechanical engineers often build from their designs physical 3D models for a better presentation, comprehension, and communication among stakeholders. Software engineers usually create visualizations of their software designs as digital objects to be presented on a screen only. 3D software visualization metaphors, such as the software city metaphor, provide a basis for exporting those on-screen software visualizations into physical models. This can be achieved by 3D-printers to transfer the advantages of real, physical, tangible architecture models from traditional engineering disciplines to software engineering.**

We present a new research perspective of using physical models to support software engineering. Furthermore, we describe four envisioned usage scenarios for physical models which provide a plethora of new research topics. To examine the benefits of our concept, we investigate the first usage scenario by evaluating the impact of using physical models on program comprehension in teams through a first controlled experiment.

Since the usage of physical models had a diverging influence for our chosen task set, we report on the qualitative results in this paper. We observed that the physical models improved the team-based program comprehension process for specific tasks by initiating gesture-based interaction.

I. Introduction

As stated by Ball and Eick [1] in 1996, "software is intangible, having no physical shape or size." However, effective program comprehension requires abstractions from source code in larger projects. To overcome this challenge, many software visualization techniques exist. Such software visualizations often rely on metaphors, such as the 3D software city metaphor [2].

Although 3D visualizations can deliver more information compared to 2D visualizations due to its additional dimension, it is often difficult for users to navigate in 3D spaces using a 2D screen and a 2D input device [3]. As a consequence, users may get disoriented [4] and thus the advantages of the third dimension may be abolished.

One solution candidate for the navigation issue is Virtual Reality (VR). It provides benefits via stereoscopy in combination with specialized hardware [5], [6]. However, it relies on – sometimes expensive – extra equipment which has to be purchased and might not function in every environment.

Traditional engineering disciplines overcome these issues by building solid, physical 3D models of their designs. Beneath resolving navigation issues, the physical models are used for better presentation, comprehension, and communication among stakeholders.

In this paper, we present a new research perspective to transfer these advantages to software engineering by building physical models following the 3D software city metaphor through 3D-printing. Furthermore, we describe four envisioned, potential usage scenarios for physical models and formulate possible research questions. To the best of our knowledge, we are the first to envision and create such physical models of software visualizations.

To show the potential of using physical models, we conduct a first controlled experiment for our first envisioned usage scenario, i.e., in the context of program comprehension. Gestures support in thinking and communication processes [7] and thus can enhance program comprehension in groups. We hypothesize that physical models support in gesticulation. In our experiment, we compare the usage of an on-screen model (visualized on a plain 2D screen) to the usage of a physical model for program comprehension in small teams (pairs). Since the overall results were not significant due to our chosen task set, we focus on the qualitative aspects of the experiment in this paper. For further additional details about the experiment, we refer to our longer technical report [8]. To facilitate the verifiability and reproducibility of our results, we provide a package [9] containing all our data including the raw data and 112 recordings of the participant sessions.

In summary, our main contributions are:

1. a new research perspective to transfer the advantages of physical models to software engineering through 3D-printing,
2. four envisioned usage scenarios for physical models and possible research questions, and
3. the results of an execution with 56 teams of a first controlled experiment comparing the usage of an on-screen model to the usage of a physical model in typical program comprehension tasks.

The remainder of this paper is organized as follows. The envisioned usage scenarios and possible research questions for physical models are described in Section II. Then, Section III introduces our virtual models and how we create physical models of them. Section IV presents the results of a controlled experiment to evaluate the impact of using physical models for program comprehension. Encountered challenges during the creation of physical models are described in Section V. Related work is discussed in Section VI. Finally, we draw the conclusions and illustrate further future work in Section VII.

978-1-4673-7526-9/15 © 2015 IEEE

II. Envisioned Usage Scenarios

Physical models provide a plethora of future research possibilities. We envision several potential usage scenarios for physical models which we will outline in the following and propose selected research questions.

A. Program Comprehension in Teams

Gestures support in thinking and communication processes [7]. Since physical models are more accessible than 2D screens and provide a natural interaction possibility, they might increase the gesticulation of users. This might lead to faster and better understanding when applied in a team-based program comprehension scenario due to its supporting nature. Furthermore, the advantages might increase when applied in larger teams. Since software systems are often changing, the model should only be printed for special occasions, e.g., a new developer team or upcoming major refactorings.

Potential Research Questions: In which scenarios/tasks do physical models provide benefits? Does the additional usage of physical models provide advantages compared to sole on-screen tooling? How large is the impact of gesticulation on correctness and time spent in team-based program comprehension tasks?

B. Educational Visualization

A further usage scenario is the usage of 3D models for educational purposes. Like an anatomic skeleton model used in a biology course, 3D models of design patterns, architectural styles, or reference architectures could be 3D-printed. Advantages include the possibly increased interest of the students and due to a 3D visualization and the possibility to touch the model, there might be a higher chance to remain in memory. Further interaction possibilities, e.g., plugging mechanisms, with the 3D model could be developed to support the learning process of the students.

Potential Research Questions: Which software visualization metaphor provides the best basis for representing design patterns? How large is the impact of using physical models for educational purposes? How to display the dynamic behavior in physical models which is often defined by a design pattern?

C. Effort Visualization in Customer Dialog

A further potential field of application are dialogs with customers. Customers often see the GUI as the program since the actual program logic code is often invisible for them. Therefore, the – possible large – effort to add a feature or to refactor the code is also often invisible for them. Presenting a physical 3D model of the status quo and another physical 3D model of the desired state, might convince the customer of the effort of the required change. This could also be achieved with two on-screen software visualizations but a touchable and *solid* 3D model might provide higher conviction.

Potential Research Questions: Do customers accept physical models to show the effort? Does the usage of physical models increase the conviction of effort in a customer dialog? How much impact does a physical model provide in this process (e.g., measured in amount of money for the effort)?

D. Saving Digital Heritage

We envision physical models to be a step toward saving the digital heritage of software visualizations. Compared to programs, physical models do not depend on the availability of SDKs, library versions, or hardware and thus are less vulnerable to changes of external environment. Often it is uncertain, if the code can still be run in thirty years. In contrast, depending on the material (e.g., resin or metal), physical models might last hundreds of years. One might argue that pictures of the visualizations are sufficient. However, they do not provide interaction possibilities and can suffer from occlusion. Contrary, physical models still provide the possibility to interact (e.g., rotate) avoiding possible occlusion.

Potential Research Questions: How to provide an omni accessible archive for physical models? How to convert 2D software visualizations to 3D physical models?

III. From Virtual to Physical Models

A. Our City Metaphor in a Nutshell

This section introduces the semantics of our city metaphor implemented in our web-based ExplorViz[1] tool. Figure 1 displays a physical model visualizing the quality tool PMD[2] utilizing our city metaphor. The visualization is constructed from an execution trace of PMD used as the object system in our controlled experiment. The flat green platforms in our visualization represent packages showing their contained elements. The green boxes on the top layer are packages hiding their internal details. The two green tones serve to differentiate the hierarchy levels. Classes are visualized by purple boxes. The height of classes maps to the active instance count. The layout is a modified version of the layout used in CodeCity [10].

B. Creating Physical 3D-Printed Models

After creating the on-screen model in ExplorViz by monitoring a PMD run, we exported the model as an OpenSCAD[3] script file. To fit the build platform of our low-budget 3D-printer (a Prusa i3), we split the exported model into twelve jigsaw pieces and exported these as STL[4] files as input for the 3D-printer. After printing each piece, we glued them together. The fully assembled physical PMD model is 334 mm wide and 354 mm deep. Finally, we manually painted the single-colored model. The overall building time of the physical PMD model (see Figure 1) sums up to 58 hours and the costs of material are about 9 € on our low-budget, self-built, and rather slow 3D-printer. On a modern multi-color printer, 3D-printing the model would take only a small fraction of the building time (i.e., a few hours) and save several of the described working steps. The encountered challenges for creating physical models are described in Section V.

[1] http://www.explorviz.net
[2] http://pmd.sourceforge.net
[3] http://www.openscad.org
[4] http://www.fabbers.com/tech/STL_Format

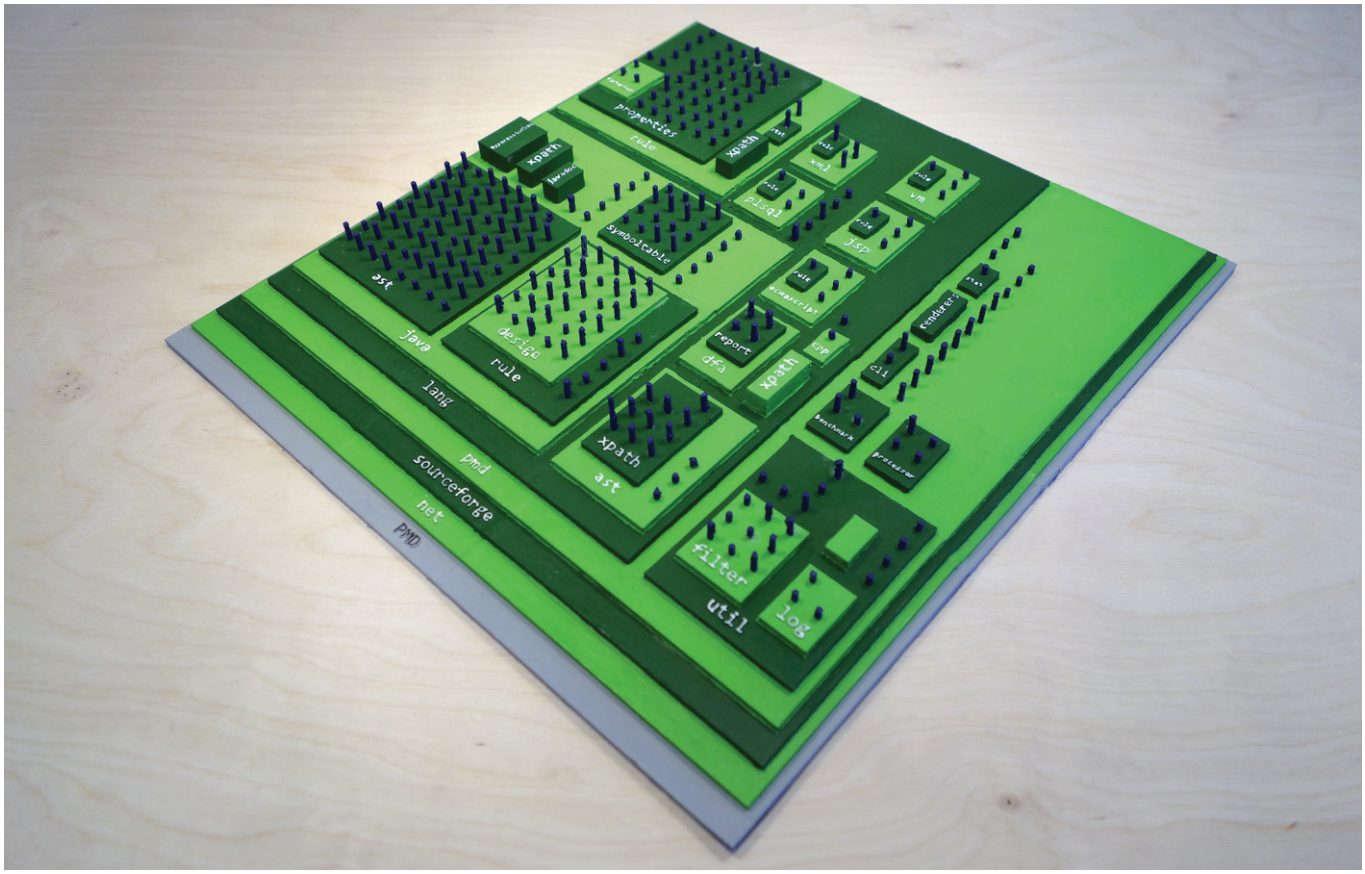

Fig. 1. Physical 3D-printed and manually painted city metaphor model of PMD (334 mm wide and 354 mm deep)

IV. First Evaluation for Program Comprehension

Aside from introducing physical models in this paper, we present the qualitative results of a first evaluation for the envisioned physical models. We compared the impact of using either an on-screen model (i.e., a virtual model) or a physical model to solve typical program comprehension tasks in a team-based scenario. As object system we used PMD and measured the *correctness* of the solutions and *time spent* for each task typically used in the context of program comprehension [11]. Afterwards, we analyzed the employed strategies and possible differences between both groups.

Since the overall results were not significant due to our chosen task set, we focus on the qualitative aspects of the experiment in this paper and only briefly describe the design. For further additional details about the experiment, we refer to our longer technical report [8].

We used a between-subjects design. Thus, each subject solved tasks with either using the on-screen or the physical model. The 112 participants – forming 56 teams – were students. The teams were assigned to the control or the experimental group by random assignment.

To summarize the impact of using physical models in our task set: Two tasks were positively influenced by the physical model. In contrast, one task was negatively influenced and the rest of the five tasks were even or without clear statement.

In the following, we report on the analysis of the two positively influenced tasks to point out beneficial scenarios for physical models. The reasons behind the one negatively influenced task (*Which package name occurs the most in the application?*) was mainly due to some less readable labels in the physical model.

The first positively influenced task was: *Assuming a good design, which package could contain the Main class of the application?* The physical model group outperformed the on-screen model group by 19% in average correctness while using roughly the same time. In the on-screen model group, the team partner often had to search for the package name that the teammate was talking about. In contrast, the physical model group often used gestures to show the package under investigation. Furthermore, all packages could be clearly seen without scrolling or rotating.

The second positively influenced task was: *What is the purpose of the* `lang` *package and what can you say about its content regarding PMD?* The on-screen model group required more time (about 10% longer) for roughly the same correctness score as the physical model group. We observed that both team partners used gestures during discussion in the physical model group - also in parallel. In contrast, it was harder for the on-screen model group to use gestures in parallel since one arm often occludes the screen. These parallel gestures could

have increased the efficiency to solve the task in the physical model group.

In summary, we observed a higher amount of gestures in the physical model group compared to the on-screen model group. These were used to communicate with the team partner and reading the package paths. Difficulties with the physical model were encountered due to less readable labels.

V. Encountered 3D-Printing Challenges

During the process of creating a physical model from a virtual city metaphor model, we encountered several challenges. In this section, we provide an overview of these and present how we overcame them to enable other researches to create physical models of their own visualizations.

A. Potential 3D-Printer Input Format Change

Since being a new technology in the consumer market, the input format for 3D-printers may still change. To mitigate this risk, ExplorViz exports OpenSCAD script files of the virtual models and not a directly 3D-printable formated file. The script files contain commands how to create entities, for instance, boxes with their position. The resulting rendered objects can be exported to six different file formats, at the moment.

B. Mapping Virtual Dimension to Physical Dimension

The virtual city metaphor model has its own virtual dimension in ExplorViz but this dimension has no concrete relation to a physical dimension. To create a physical model, we had to find a mapping coefficient.

There are two opposing factors. The smallest parts are the buildings (classes) and the labels. Both should be printed as large as possible to avoid fragile buildings and to ensure readability of the labels. On the opposite, the overall model should be as small as possible since a huge model would require more head movement and time to grasp the model. After a period of prototyping, we found a mapping coefficient which is a trade-off between both factors.

C. Creating Labels

To provide a useful physical model, the components had to be labeled. Since June of 2014, OpenSCAD directly supports rendering fonts via the `text()` command. However, some fonts are harder to print for the 3D-printer and are less readable at small size. Therefore, we tested some fonts and achieved the best performance with *Consolas*.

D. Limited Build Volume

We use a Prusa i3[5] to print our city metaphor models due to its rectangular build platform. It can print objects with a size of up to 200 mm^3. However, our models can easily get larger than this volume, as in the case of the PMD model presented in Figure 1. Therefore, we split the models into multiple smaller jigsaw pieces using the PuzzleCut[6] library for OpenSCAD.

[5]http://reprap.org/wiki/Prusa_i3

[6]http://nothinglabs.blogspot.de/2012/11/puzzlecut-openscad-library.html

Fig. 2. One unpainted jigsaw piece of our physical PMD model

Figure 2 shows one jigsaw piece from our PMD model. After the print, the model is assembled and agglutinated.

E. Monochromacity

Most of the current consumer 3D-printers, which use the fused filament fabrication technique, create only single-colored objects. Three-colored objects are possible but the 3D-printer must be upgraded. Our city metaphor models contain six colors and thus they require a different solution at the moment. We prime the printed model using a white spray can. Afterwards, we use miniature figure colors to manually paint the models. An unpainted model and some used colors can be seen in Figure 2.

VI. Related Work

Since we are – to the best of our knowledge – the first creating physical models of software visualizations, there is only related work to our visualization and evaluation methodology. However, general information visualization also uses 3D-printing to create physical visualizations.[7] First, we differentiate us from related work concerning the city metaphor. Afterwards, we describe virtual reality approaches, which also aim for a more natural impression and intuitive navigation. Finally, software visualization experiments, which compare themselves to other software visualizations, are discussed. Since out of our focus, we do not discuss experiments comparing software visualizations to IDEs and refer to [10] for more details.

A. City Metaphor

In contrast to existing city metaphor approaches, e.g., [2], [10], [12], we enable the user to create physical models from the on-screen presentation to facilitate, for instance, more intuitive navigation.

[7]http://dataphys.org/list

B. Virtual Reality for Software Visualization

Imsovision [13] aims at representing object-oriented software in a virtual reality environment. The user is tracked with electromagnetic sensors attached to the shutter glasses and a *wand* which is used for the 3D navigation.

SykscrapAR [14] is an augmented reality approach employing the city metaphor to visualize software evolution. The user can interact with a physical marker platform in an intuitive way while the actual visualization can be seen only on the monitor.

Contrary to both approaches, our physical models do not require any additional devices once they are created.

C. Experiments Comparing Software Visualizations

Storey et al. [15] compared three software visualizations in an experiment. The authors performed a detailed discussion of the tools' usage but provided no quantitative results.

Lange and Chaudron [16] investigated the benefits of their enriched UML views by comparing them with traditional UML diagrams. In contrast, we compare the impact of using physical models on program comprehension.

VII. Conclusions And Outlook

In this paper, we presented the vision of transferring the advantages of physical, tangible models to support software engineering. We described four potential usage scenarios, and investigated the impact of using physical models on team-based program comprehension by a first controlled experiment.

In our experiment, we identified two tasks that benefit from using physical models in comparison to using on-screen models. However, our experiment resulted in no overall impact neither on time spent nor on correctness of solutions to the program comprehension tasks, since the effects of each task compensate each other for our chosen task set.

Our in-depth analysis of the strategies used by the teams supports our hypothesis that physical models provide an appropriate, complementary communication basis and increase interaction when solving comprehension tasks in small teams. In the analysis, we observed an increase in the amount of performed gestures when using the physical model.

We provide a package containing all our experimental data to facilitate the verifiability and reproducibility for replications and further experiments. It contains the employed version of ExplorViz v0.6-exp (including source code and manual), input files, STL files for 3D-printing the used models, tutorial materials, questionnaires, R scripts, dataset of the raw data and results, and 112 screen and camera recordings of the participant sessions. The package is available online [9] with source code under the Apache 2.0 License and the data under a Creative Commons License (CC BY 3.0).

In the context of program comprehension, future work is manifold. To validate our observations in the controlled experiment for program comprehension, further replications should be conducted. Furthermore, our experiment design should be tested with a larger team size where gesticulation and communication can have a higher impact, and with professionals as subjects. Further experiments should examine the combination of using a physical model and an on-screen model compared to solely using an on-screen model. Likewise, physical models should be compared to Virtual Reality and could be enhanced by Augmented Reality.

On a more general perspective, we only investigated a part of the first usage scenario of the four envisioned scenarios revealing a promising future research direction. The other scenarios should also be evaluated. Another direction is the creation of physical models of other 3D software visualization metaphors, e.g., trees [17], spheres [18], or solar systems [19]. Every usage scenario might reveal different results with other representation of physical models.

References

[1] T. Ball and S. G. Eick, "Software visualization in the large," *Computer*, vol. 29, no. 4, pp. 33–43, 1996.

[2] C. Knight and M. Munro, "Virtual but visible software," in *Proc. of IEEE Int. Conf. on Inf. Vis. (IV 2000)*. IEEE, 2000, pp. 198–205.

[3] A. Teyseyre and M. Campo, "An overview of 3D software visualization," *IEEE TVCG*, vol. 15, no. 1, pp. 87–105, Jan. 2009.

[4] K. P. Herndon, A. van Dam, and M. Gleicher, "The challenges of 3D interaction: A CHI '94 Workshop," *SIGCHI Bull.*, vol. 26, no. 4, pp. 36–43, Oct. 1994.

[5] C. Ware, K. Arthur, and K. S. Booth, "Fish tank virtual reality," in *Proceedings of the INTERACT 1993 and Conference on Human Factors in Computing Systems (CHI 1993)*. ACM, 1993, pp. 37–42.

[6] C. Ware and P. Mitchell, "Reevaluating stereo and motion cues for visualizing graphs in three dimensions," in *Proc. of 2nd Symp. on Applied Perc. in Graph. and Vis. (APGV 2005)*. ACM, 2005, pp. 51–58.

[7] S. Goldin-Meadow, *Hearing gesture: How our hands help us think*. Harvard University Press, 2005.

[8] F. Fittkau, E. Koppenhagen, and W. Hasselbring, "Research perspective on supporting software engineering via physical 3D models," Kiel University, Tech. Rep. 1507, Jun. 2015.

[9] ——, "Experimental data for: Research perspective on supporting software engineering via physical 3D models," Jun. 2015, zenodo.org. doi: 10.5281/zenodo.18378.

[10] R. Wettel, M. Lanza, and R. Robbes, "Software systems as cities: A controlled experiment," in *Proc. of 33rd Int. Conf. on Software Engineering (ICSE 2011)*. ACM, 2011, pp. 551–560.

[11] V. Rajlich and G. S. Cowan, "Towards standard for experiments in program comprehension," in *Proc. of 5th Int. Workshop on Program Comprehension (IWPC 1997)*. IEEE, 1997, pp. 160–161.

[12] T. Panas, R. Berrigan, and J. Grundy, "A 3D metaphor for software production visualization," in *Proc. of 7th Int. Conf. on Information Visualization (IV 2003)*. IEEE Computer Society, 2003, pp. 314–320.

[13] J. I. Maletic, J. Leigh, A. Marcus, and G. Dunlap, "Visualizing object-oriented software in virtual reality," in *Proc. of 9th Int. Workshop on Prog. Comprehension (IWPC 2001)*. Society Press, 2001, pp. 26–35.

[14] R. Souza, B. Silva, T. Mendes, and M. Mendonca, "SkyscrapAR: An augmented reality visualization for software evolution," in *Proc. of 2nd Brazilian Workshop on Software Visualization (WBVS 2012)*, 2012.

[15] M.-A. Storey, K. Wong, and H. Müller, "How do program understanding tools affect how programmers understand programs?" in *Proc. of 4th Work. Conf. on Reverse Eng. (WCRE 1997)*. IEEE, 1997, pp. 12–21.

[16] C. Lange and M. R. V. Chaudron, "Interactive views to improve the comprehension of UML models – An experimental validation," in *Proc. of 15th Int. Conf. on Prog. Comp. (ICPC 2007)*, June 2007, pp. 221–230.

[17] E. Kleiberg, H. Van De Wetering, and J. J. Van Wijk, "Botanical visualization of huge hierarchies," in *IEEE Symposium on Information Visualization*. IEEE Computer Society, 2001.

[18] M. Balzer, A. Noack, O. Deussen, and C. Lewerentz, "Software landscapes: Visualizing the structure of large software systems," in *Proceedings of the Sixth Joint Eurographics-IEEE TCVG conference on Visualization*. Eurographics Association, 2004, pp. 261–266.

[19] H. Graham, H. Y. Yang, and R. Berrigan, "A solar system metaphor for 3D visualisation of object oriented software metrics," in *Proceedings of the Australasian Symposium on Information Visualisation (APVIS 2004)*. Australian Computer Society, Inc., 2004, pp. 53–59.