International Journal of
Emerging Research in Management &Technology
ISSN: 2278-9359

Research Article | **January 2013**

Software Architecture Visualization

Jyothi Priya*
Dept of Computer Applications
CMJ University
Bangalore, India

Dr. Mohammad Sharif K B
Dept of Computer Applications
CRD Business School
Bangalore, India

Badri H.S
Dept of Computer Applications
Presidency College
Bangalore, India

Abstract:

Understanding the software architecture is a vital step towards building and maintaining software systems. But software architecture is an intangible conceptual entity. Therefore, it is hard to comprehend software architecture without a visual mapping that reduces the burden on the human brain. Visualizing software architecture has been one of the most important topics in software visualization. Not only are architects interested in this visualization but also developers, testers, project managers and even customers. This paper is a survey on recent and key literature on software architecture visualization. It touches on efforts that defined what characteristics an effective visualization should have. It compares various efforts in this discipline according to taxonomies such as dimensionality, multiplicity of views and use of metaphors. The paper also discusses trends and patterns in recent research and addresses research questions that are still open for further investigation

Keywords: Software, Software Architecture, Software Systems, Visualization, Taxonomies

I. INTRODUCTION

Simply put, Software Visualization (SV) is the use of visual representations to enhance the understanding and comprehension of the different aspects of a software system. Price et al. [1] gives a more precise definition of software visualization as the combination of utilizing graphic design and animation combined with technologies in human-computer interaction to reach the ultimate goal of enhancing both the understanding of software systems as well as the effective use of these systems. The need to visualize software systems evolved from the fact that such systems are not as tangible and visible as physical objects in the real world [2]. This need becomes particularly evident when the software system grows to entail a huge number of complexly related modules and procedures. This growth results in a boost in the time and effort needed to understand the system, maintain its components, extend its functionality, debug it and write tests for it. SV is a broad field that is concerned with visualizing aspects of software engineering as a practice and software systems as evolving products. Some of these aspects include design models and patterns, software architecture, development processes, code history, database schemes, network interactions, web services, parallel processing, process execution and many others [3]. This paper focuses on a single aspect of SV which is software architecture. Software architecture refers to the structure of a software system including composing entities, the metrics of these entities, and the relationships among different entities [4]. Visualizing software architecture encompasses not only the software modules and their internal structures and interrelations, but also the evolution of these modules over time [5]. Considering the numerous tools and approaches proposed to directly or indirectly help build an understanding of the system architecture (including software exploration tools), research in this area has been extensive especially in the last few years.

II. QUESTIONS & PROBLEMS

In this section, we touch on the main problems and questions recent research has been trying to tackle. Research in the area of software architecture visualization is centered on finding a meaningful and effective mapping scheme between the software architecture elements and visual metaphors [6]. Recent research has been trying to answer different questions such as: “why is the visualization needed?”, “who will use [it]?”, and “how to represent it?” [7]. Others like [8] questioned the effectiveness and expressiveness of the visuals to use. In general the various questions asked in this discipline can be grouped into three broad categories: • Who are the different groups of audience for architecture visualization [9]? • What questions do they wish to answer through this visualization [10]? • How can visual metaphors and interaction techniques be used to answer their questions
[11]?

A WHO?.

By answering the “who” question, we determine the different groups of audience for architecture visualization and their various interests. Defining the audience of the architecture visualization plays a pivotal role in determining what to visualize and how to visualize it. McNair et al. [9] defines three different groups that are interested in the software architecture of a given system. These three groups are developers, managers and researchers. Panas et al. [10] consider architects to be one additional group, and they differentiate between developers and maintainers.

B WHAT?.

Having defined the audience for architecture visualization, it is equally important to understand the needs of this audience. These needs can be understood by studying the roles different groups of audience play and the influences they have on the software architecture. The questions these groups address differ according to their various interests and level of involvement in the software project. Therefore, the interests of these groups are to be considered when visualizing software architecture. For example, it is important for architects to realize the different characteristics of the architecture they design such as complexity, coupling, cohesion and other attributes. Developers (and maintainers) are usually interested in understanding how the architecture is laid out, and what the most recent status of the software system is. Developers deem this understanding important
because it enables them to maintain the system and continue the development process. Managers; on the other hand, want, on a high level, to be able to monitor the progress of the project and determine the completion of development goals. On another level are researchers. They are interested in studying characteristics and trends of software architectures and their evolution in general. Customers are also considered an important audience for architecture visualization [12]. They might ask or need to have a general overview about the design of the system, but they need not look at low level details that are irrelevant to their concerns. Figure 1 illustrates the previously mentioned groups along with their interests in the various levels of detail in the software architecture based on [9], [10] and [12].

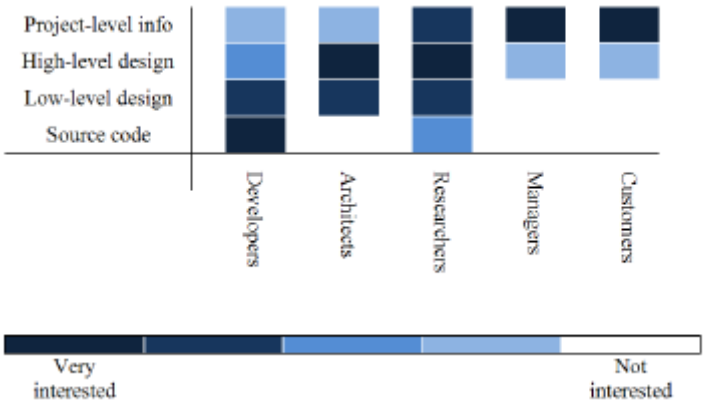

Figure 1 - Various levels of interest of SV audience

C HOW?.

To satisfy the needs of the different stakeholders, the visualization needs to be capable of answering their various questions about the software architecture. Numerous approaches, techniques and tools have been proposed by researchers in an attempt to produce an effective visualization that can achieve this goal. But what an effective visualization is can be an ambiguous issue that inevitably might create a gap between the proposed solutions and the needed ones. This issue of determining the effectiveness of the visualization has driven a new stream of research dedicated to answering the question: "What makes architecture visualization effective?" The following section will discuss some of the efforts to establish reliable and valid criteria that determine the effectiveness of a proposed solution.

III. SOLUTIONS: TOOLS & TECHNIQUES

Efforts in the field of architecture visualization can be categorized in more than one way. The following is a comparative review of existing solutions based on three taxonomies: multiplicity of view, dimensionality and metaphor. In the following survey, these characteristics will overlap resulting in some papers being mentioned more than once under different sections.

A Multiplicity of view.

With regards to the multiplicity of view, two schools of thoughts can be identified. On the one hand, the first school asserts that any visualization should support multiple views of the architecture at different levels of detail in order to satisfy the audience's different interests [18, 21, 22]. That is, for the visualization to be deemed useful, it has to provide a means of looking at the different aspects of a software architecture through different views, and possibly via multiple windows. For instance, if one view provides an insight into the internal structure of software entities composing the architecture, another view should, on a higher level, focus on the relationships and communication between these entities. The other school of thought; on the other hand, believes that a carefully designed single view of the visualization might

be more effective and meaningful in conveying the multiple aspects of the architecture than the multiple view approach [10, 14]. For example, the tool can provide different levels of detail in a single view (e.g. internal structures of entities along with the relationships between them) and leave it up to the viewers to draw their own mental maps at the level of interest to them.

1) Multiple-view visualization:

Lanza et al. [23, 24] introduced a software visualization tool called CodeCrawler (CC). Through a 2D visualization of reverse-engineered object oriented software systems, CC offers the advantage of having multiple views of the same architecture. The multiplicity of views aims to uncover the different aspects of the architectural design and emphasize specific metrics in the software system. In general, CC represents the architecture in a polymeric view in which entities (classes, methods … etc) are represented as nodes and relationships as edges connecting these nodes. The node's size, position and color are used to represent the metrics of interest in the software system. There are four different views that CC has to offer: 1) coarse-grained view by which the system complexity is emphasized, 2) finegrained view which hierarchically represents the class blueprint in the architecture, 3) coupling view which, as the name suggests, underscores coupling amongst modules in the architecture, and 4) evolutionary view which helps track changes of the architecture over time. Figure 2 shows the four different views. Lanza et al.
explain, in great detail, the algorithm followed to layout the class blueprint visualization in a separate paper [25].

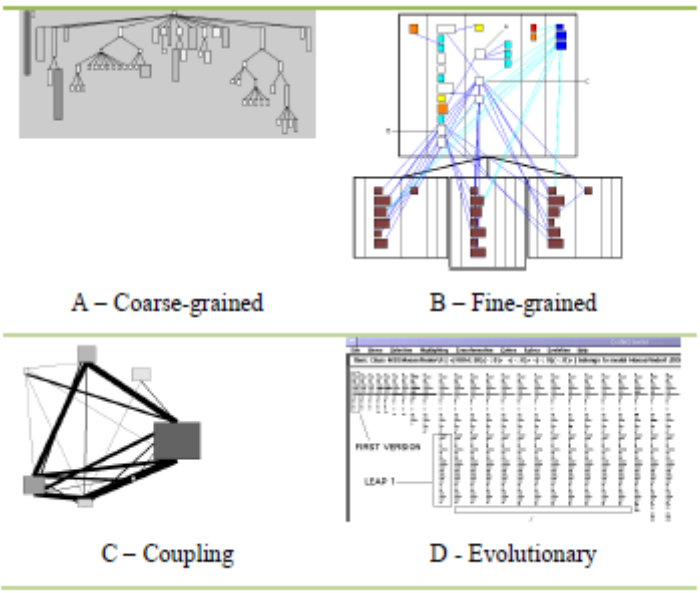

Figure 2 - Four different views for the software architecture by Lanza et al. [23, 24]

IV. TRENDS

In this paper, the main focus of our literature review was research efforts in the last few years. It is indeed difficult to make conclusions about trends in the discipline of software architecture visualization without also looking at research efforts in the last two decades. This is because the incremental advancements in this field, as you might have noticed from the previous sections, are not significant enough to notice revolutionary changes in a short period of time. Actually, some works are too similar to be able to tell what the significance of the latter work is. For example, in 2007, the "habitability" or the "city" metaphor has been a repeatedly proposed visualization metaphor with very small incremental enhancements [10, 11, 35]. For these reasons, the reader is asked to keep in mind, when reading the analysis in this section, that little context has been considered before the year 2001. The plot in Figure 2 shows some of the papers in the architecture visualization over a 7 year period. The x-axis represents the years whereas the y-axis indicates the references to these works in our paper. A box can be either white (2D), grey (3D) or black (VE). A dotted frame indicates multiple-view whereas a connected frame indicates single-view. Each box contains either the letter "A" for abstract metaphor or "R" for real metaphor. As seen in Figure 2, the most obvious trend in software architecture visualization (SAV) is the use of real metaphors as opposed to abstract ones. By 2005, most of the proposed efforts focused on delivering visualizations that make use of real-life objects to represent software entities. This trend has been most probably supported by

the advancement in graphics-related technologies (software and hardware) rather than empirical evidence of the advantages of using a real metaphor in software visualizations. Literature lacks empirical evaluation of what the added benefit is when utilizing a "city" metaphor for example. There have been evaluations that tested utilization of space and other visual attributes, but none (from the set covered) touched on the cognitive aspect of the effectiveness of these real metaphors.

Adding an extra dimension for existing 2D visualization solutions has also been a noticeable trend in recent literature. 3D visualizations became more appealing after the advantages of 3D visualizations over 2D ones were uncovered through practical comparisons [16]. Although in 2005, there were still some proposed 2D solutions, we can notice that in 2007, most (if not all) of the proposed solutions consider the third dimension. This transition to 3D spaces can be deemed an enhancement (as opposed to the previous trend of real metaphors) because of the accompanying (old and new) evidences of what the added benefits are. Considering the third taxonomy, there is no obvious trend other than the original practice of providing multiple views of the architecture to underscore the different design aspects. Some people did suggest that single-view visualizations are more effective, but their argument was not backed by any experiments or empirical studies.

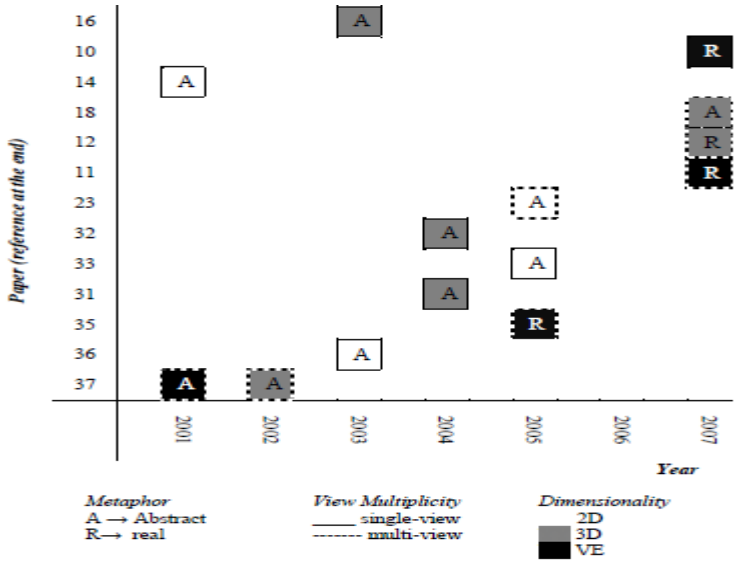

Figure 2 - Some papers on SAV from 2001 to 2007

V. CONCLUSION

Software visualization is a broad topic that has increasingly been extended to cover many aspects of software engineering and software systems. Two main conferences lead research in this area: "The ACM Symposium on Software Visualization (SOFTVIS)" and "The IEEE International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT)". Other conferences like SIGGRAPH and INFOVIS are also valuable venues for this discipline. This paper talked about a specific area in software visualization concerned with visualizing software architecture. To begin with, we defined the field and its position within the scope of software visualization. Then, we discussed the three main axes around which all efforts in this field orbit, namely: the audience, their needs and the tools and techniques to satisfy these needs. After that, we mentioned the attributes against which, as researchers suggested, visualization efforts should be evaluated such as expressiveness, user interaction, visual complexity and navigation techniques.

ACKNOWLEDGMENT

I am indebted to Dr. Mohammad Sharif Bammanalli for his valuable insights and guidance.

REFERENCES

[1] Price, B.A., Baecker, R., and Small, I.S. (1998) A principled taxonomy of software visualisation. *Journal of Visual Languages and Computing*. 4(3), pp. 211- 266.

[2] Petre, M., and de Quincey, E. (2006) A gentle overview of software visualization. *The Computer Society of India Communications (CSIC) & Autumn 2006 PPIG newsletter*.

[3] ACM Symposium on Software Visualization. Available at http://www.st.unitrier. de/~diehl/softvis/org/softvis08, last accessed Feb 20, 2008.

[4] L. Bass, P. Clements, and R. Kazman. (2003) *Software Architecture in Practice*. Addison – Wesley Inc.
[5] D'Ambros, M., and Lanza, M. (2007) BugCrawler: Visualizing Evolving Software Systems. *The 11th European Conference on Software Maintenance and Reengineering, 2007*, pp.333-334.
[6] Gračanin, D., Matković, K., and Eltoweissy, M. (2005) Software visualization. *Innovations in Systems and Software Engineering*, 1(2), pp. 221-230, Springer London.
[7] Maletic, J., Marcus, A., and Coollard, M. (2002). A Task Oriented View of Software Visualization. *IEEE Workshop of Visualizing Software for Understanding and Analysis, 2002*, pp. 32-40.
[8] Mackinlay, J. (1986) Automating the design of graphical presentation of relational information. ACM Transaction on Graphics, 5(2), pp. 110-141.
[9] McNair, A., German, D., Weber-Jahnke, J. (2007) Visualizing Software Architecture Evolution Using Change-Sets. *The 14th Working Conference on Reverse Engineering, 2007*, pp.130-139.
[10] Panas, T., Epperly, T., Quinlan, D., Saebjornsen, A., and Vuduc, R. (2007) Communicating Software Architecture using a Unified Single-View Visualization. *The 12th IEEE International Conference on Engineering Complex Computer Systems, 2007*, pp. 217-228.
[11] Alam S., and Dugerdil P. (2007) EvoSpaces: 3D Visualization of Software Architecture. *The 19th International Conference on Software Engineering & Knowledge Engineering, 2007*.
[12] Boccuzzo, S., and Gall, H. (2007) CocoViz: Towards Cognitive Software Visualizations. *The 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis, 2007*, pp.72-79.
[13] Storey, M., Fracchia, F., and Muller, H. (1997) Cognitive design elements to support the construction of a mental model during software visualization. *The Fifth International Workshop on Program Comprehension, 1997*, pp.17-28.
[14] Storey, M., Best, C., and Michand, J. (2001) SHriMP views: an interactive environment for exploring Java programs. *The 9th International Workshop on Program Comprehension, 2001*, pp.111-112.
[15] Storey, M., Best, C., Michaud, J., Rayside, D., Litoiu, M., and Musen, M. (2002) SHriMP views: an interactive environment for information visualization and navigation. In *CHI '02 Extended Abstracts on Human Factors in Computing Systems, 2002*.
[16] Marcus, A., Feng, L., and Maletic, J. (2003) 3D Representations for Software Visualization. *The 1st ACM Symposium on Software Visualization, 2003*, pp. 27-36.
[17] Tilley, S. and Huang, S. (2002) Documenting software systems with views III: towards a task-oriented classification of program visualization techniques. *The 20th Annual international Conference on Computer Documentation*, 2002, pp. 226-233.
[18] Sawant, A., and Bali, N. (2007) DiffArchViz: A Tool to Visualize Correspondence between Multiple Representations of Software Architecture. *The 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis, 2007*, pp.121-128.
[19] Li, W., Eades, P., and Hong, S. (2005) Navigating software architectures with constant visual complexity. IEEE Symposium on Visual Languages and Human- Centric Computing, 2005, pp. 225-232. 20. Young, P., and Munro, M. (2003) Visualising software in virtual reality. *IEEE First International Workshop on Visualizing Software for Understanding and Analysis, 2003*.
[20] Young, P., and Munro, M. (2003) Visualising software in virtual reality. *IEEE First International Workshop on Visualizing Software for Understanding and Analysis, 2003*.
21] Reiss, S., and Renieris, M. (2003) The BLOOM Software Visualization System. *Software Visualization – From Theory to Practice*, MIT Press, 2003.
[22] Kruchten, P. (1995) The "4+1" view model of architecture. *IEEE Software*, 12(6), pp. 42-50. 23. Lanza, M., Ducasse, S., Gall, H., and Pinzger, M. (2005) CodeCrawler - an information visualization tool for program comprehension. *The 27th International Conference on Software Engineering, 2005*, pp. 672-673.
[23] Lanza, M., Ducasse, S., Gall, H., and Pinzger, M. (2005) CodeCrawler - an information visualization tool for program comprehension. *The 27th International Conference on Software Engineering, 2005*, pp. 672-673.
[24] Lanza, M. (2003) CodeCrawler - A Lightweight Software Visualization Tool. *The 2nd International Workshop on Visualizing Software for Understanding and Analysis, 2003*, pp. 51 - 52.
[25] Lanza, M., and Ducasse, S. (2001) The Class Blueprint - A Visualization of the Internal Structure of Classes. *Software Visualization Workshop (OOPSLA, 2001).*
[26] Michaud, J., Storey, M., and Muller, H. (2001) Integrating information sources for visualizing Java programs. *IEEE International Conference on Software Maintenance, 2001*, pp.250-258.
[27] Lintern, R., Michaud, J., Storey, M., and Wu, X. (2003) Plugging-in visualization: experiences integrating a visualization tool with Eclipse. *The ACM Symposium on Software Visualization, 2003*, p. 47.
[28] Voinea, L., Lukkien, J., and Telea, A. (2007) Visual assessment of software evolution. *Science of Computer Programming*, pp. 222-248.
[29] Ware, C., Hui, D., and Franck, G. (1993) Visualizing object oriented software in three dimensions. *The IBM Center for Advanced Studies Conference, 1993*, pp. 612-660.

[30] Jones, J., Harrold, M., and Stasko, J. (2001) Visualization for Fault Localization. *The ICSE Workshop on Software Visualization, 2001*, pp. 71-75.
[31] Balzer, M., Noack, A., Deussen, O., and Lewerentz, C. (2004) Software landscapes: Visualizing the structure of large software systems. *The ACM Symposium on Software Visualization, 2004*, pp. 261-266.
[32] Balzer, M., and Deussen, O. (2004) Hierarchy Based 3D Visualization of Large Software Structures. *IEEE Visualization, 2004.*
[33] Balzer, M., Deussen, O., and Lewerentz, C. (2005) Voronoi treemaps for the visualization of software metrics. *The ACM Symposium on Software Visualization, 2005*, pp. 165-172.
[34] Lowe, W., and Panas, T. (2005) Rapid Construction of Software Comprehension Tools. *International Journal of Software Engineering & Knowledge Engineering*, 15(6), pp. 905-1023.
[35] Wettel, R., and Lanza, M. (2007) Program Comprehension through Software Habitability. *The 15th IEEE International Conference on Program Comprehension, 2007*, pp.231-240.
[36] Ham, F. (2003) Using Multilevel Call Matrices in Large Software Projects. *IEEE Symposium on Information Visualization, 2003*, p. 29. 37. Lewerentz, C., and Simon, F. (2002) Metrics-based 3D visualization of large object-oriented programs. *The First International Workshop on Visualizing Software for Understanding and Analysis, 2002*, pp. 70-77.
[37] Lewerentz, C., and Simon, F. (2002) Metrics-based 3D visualization of large object-oriented programs. *The First International Workshop on Visualizing Software for Understanding and Analysis, 2002*, pp. 70-77
[38] Knodel, J., Muthig, D., Naab, M., and Zeckzer, D. (2006) Towards Empirically Validated Software Architecture Visualization. IESE-Report 071.06/E.