Interactive Hyperbolic Tree for Industrial size Software Product line Architecture

Abeer Khalid, Salma Imtiaz
Department of Software Engineering
International Islamic University
Islamabad, Pakistan
abeer.msse234@iiu.edu.pk, salma.imtiaz@iiu.edu.pk

***Abstract*—This Software Product line is an eminent part of software re-engineering field. Facilitation of software product line architecture with a more convenient method of representation mechanism results in efficiency with respect to time, cost, energy, etc. For this to be true, there is a need for information visualization techniques that represent true characteristics of software product line. This paper presents a study of information visualization technique which makes perception of data easy for interacting with the software product line architecture.**

Keywords-software product line architecture; information visualization; visualization representation

I. INTRODUCTION

Software product lines are known as a family of software systems, based on common and varying aspects of software products with immense complexity rooted in them. The present studies have suggested that architecture is the best suitable form there representation [24] [25]. Literature shows that representation mechanism, such as Unified Modeling Language (UML), matrix tables, conventional trees have so far been used in illustration of software product line architecture. But foremost, they have not depicted the characteristics of a software product line, which consequences in, not well attained results. For this problem to be tackled, an information visualization technique is the best suitable option [26].

In recent years, information visualization has taken grip of software engineering field by its sheer capability to enhance cognitive abilities for perceiving complex data [23]. Thought information visualization is a relatively new concept in the branch of software product line engineering, it can still be of immense help if a suitable visual structure plus its interactive visualization techniques are provided, as well said by Tufte "There are right ways and wrong ways to show data; there are displays that reveal the truth and displays that do not" [22].

A lot of work has been done in representation of software product line architecture data, with each technique having its pros and cons. The techniques presented so far are not scalable, traceable and they are not supporting evolution [10]. Present representation mechanisms for management of software product line architecture are not capable of handling the software product line architecture attributes and do not expose good visual structure attributes [26]. And thus, a visual structure technique is proposed, which is capable of conquering the attributes of a software product line architecture data, also that visual structure can be interacted upon; without being a static structure.

Hyperbolic trees are the visual structure devising the central piece for our Information visualization techniques. The criterion, on the bases of which hyperbolic tree structure was concluded as best fit structure, was obtained from attributes of software product line architecture and visual structure [26]. The criteria were set as "abstraction, hierarchy, traceability, scalability, evolution, visual content, and perception" [26]. Also, hyperbolic trees are chosen, for the fact that they "support exponential growth in the number of components with increasing radius" [5]. Hyperbolic tree stands on the basis that it has its root in the middle while its linked nodes and their children are spread apart. In short, this hierarchy depicts many generations of parents, their children, their siblings, in the same window snapshot without losing focus of the context [6]. The main feature of hyperbolic trees is their ability to be manipulated, without any regard to its extremely large hierarchy, which is much larger than conventional hierarchal structure. They have the ability to show 10 times as many nodes compared to other visual structures, and hyperbolic tree structure being more effective in providing navigation, without deviating from the context [5]. This takes care of our software product line architecture scalability issue to some extent.

This paper is organized in four sections: Section II is concerned with the problem and related work. Section III describes the visualization of the chosen visual structure. Section IV states the conclusion and direction for future work.

II. PROBLEM AND RELATED WORK

So far, representation of software product line architecture has used many techniques and notations (e.g., Matrix table conventional tree, then notations like UML, etc.). But, noticeably all these techniques are lacking in one way or another.

Literature suggests that a number of illustration mechanisms are used for representation of software product line data. Unified modeling language (UML) notations are a well-known representation form, and can be understood easily, with platform independence provided in them[16-21]. UML notations incorporated with natural languages are also used for representation of software product line data. Use case map path notations (UCM) are also used for representation of software product line data. The point to be notated is that all of the notations are good in some context [26], but they are not favorable for representation of software

 ISBN: 978-1-61208-304-9

product line architecture data as a whole, where traceability links need to be visualized across the architecture as a whole, beside other factors.

Textual presentation is another representation form, which is used for SPL data [13] [14] [15]. But again, it is not feasible for the fact that, it is not scalable, no traceability links are present or visualized, keeping in mind that if no traceability, then evolution cannot be optimally utilized.

Matrix form is another type of notation which is used, as the literature suggests, for representation of software product line architecture data [9] [11] [12]. They are a good form of representation, but the problem with them is that they are not scalable for software product line architecture data; also, as with the above type of notation, traceability links are not visible.

Conventional trees are another type of representation form, whether they are vertical or horizontal tree [7], [9], [10]. They are the best form of presenting software product line architecture data. Here, the traceability links can be visualized for the whole context. However, they are not feasible because they are not a scalable structure, and also, when focusing on one aspect of the tree, the other parts of the hierarchy are obscured.

Cone tree is another form of hierarchal structure, which in 3D format is quite good; they overcome the prominent issues of the software product line architecture, namely scalability, plus visualization of traceability links [8], [9]. But, the problem of data obscuring is still present, meaning when focusing on one aspect of hierarchy, one does not see the full context in a single snapshot.

Tree maps are another form of hierarchal structure, which optimally utilize the screen space [7]. But the problem with this type of technique is that traceability links are not visible, also specifically one area of hierarchy cannot be focused on, without losing the grip on the context.

In sum, the shortfall of the above mentioned representation mechanism can be atoned by hyperbolic tree structure, based on the fact that its essence is favorable for software product line architecture data [26].

III. VISUALIZATION OF HYPERBOLIC TREE

The mapping of software product line architecture data on to hyperbolic tree is based on the fact that this visual structure is best suited for this job [10]. As defined in [5] and [6], hyperbolic trees support large hierarchies and their results have shown a preference towards the hyperbolic tree, as compared to conventional approaches. The authors of [5] and [6] also briefed about the implementation and the general features of their hyperbolic browser.

Here, their work has been translated for software product line architecture with enhancements included in it, based on the lack of presence of characteristics of software product line architecture. Also, the enhancements are derived from the perception capability of a human mind.

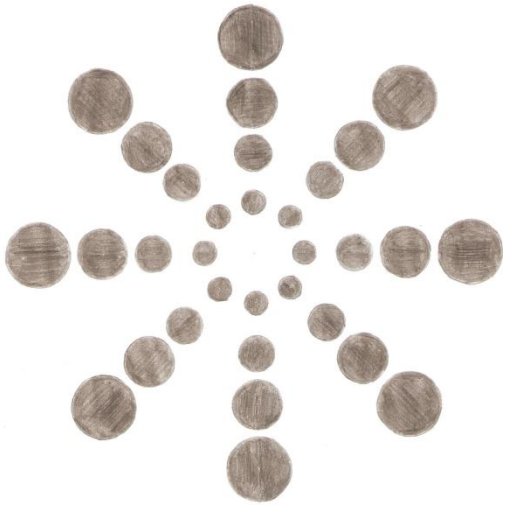

Figure 1. Based on Anstis (1974) work [3].

A. *Presenting "node"*

Each node is encompassed in a circle for displaying node information [5]. The circle does not interact with the circle of another node. The size of the circle would vary based on its generation level, e.g., if the node central to the core has size of 15cm, then, the next ring of nodes would have node with size of 10cm, which is 5cm short as compared to the parent and so on. The theory behind this logic is to show the distance factor giving the illusion of 3D depth factor. This is similar to the implementation in [3], where letter size is larger if the generation level is high. As shown in fig. 1, where outer most circle have large sized nodes, giving the perception that they are more close to the surface of the screen as compared to the other nodes; the illusion is that the size of the node decreases as they move further away from the surface of the screen. In fig. 2, Anstis [3] work has been translated onto the hyperbolic tree structure, where the inner most circles of nodes is giving the perception that they are closer to the surface of the screen. The next levels of generation of circle of nodes are positioned behind and so on.

When focusing on some point of a hierarchy then, the size of the nodes would vary, depending on the size of the parent node. The size of the parent node, and its child, and so on would become the same as compared to the other nodes at that specific time. Moreover, the positioning of the nodes with regards to the generation level would not be hindered when focusing on some part of the hierarchy.

 ISBN: 978-1-61208-304-9

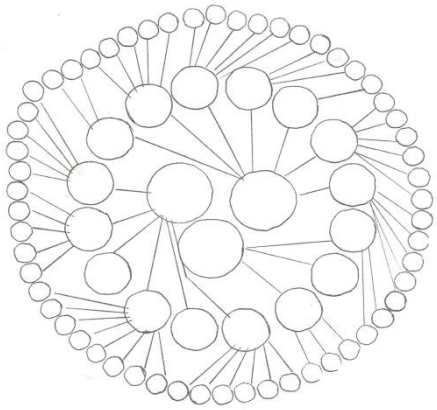

Figure 2. Hyperbolic tree structure with distance factor mapped on it.

B. Generation level

This feature has to be maintained for the sole reason that the perception of data for software product line architecture has a major hold. If the graph cannot maintain the level of placement of every node by their generation radius, then perceiving can be made quite difficult. The allotment of placement of nodes can be calculated by

$$\frac{\theta}{n} = x^{\circ} \Rightarrow x^{\circ} * s_i = y_j^{\circ} . \quad (1)$$

where θ stands for total degree of angle, n implies total number of children, x° is the equal number of angle, s_i is number of children per node, and y_j° is the number of angle per node. Also, it should be stated that for each ring of nodes this equation is called for placement of next level of generation nodes. Then [5] presented in their article, equation for calculating the needed space from a parent to their child.

Lamping and Rao formula:

$$d = \sqrt{\left(\frac{(1-s^2)\sin(\alpha)}{2s}\right)^2 + 1} - \left(\frac{(1-s^2)\sin(\alpha)}{2s}\right) \quad (2)$$

where α is angle between midline and edge of the subwedge and s is the desired distance between child and edge of its subwedge [5].Keeping in mind that even when focusing on some part of the hierarchy the level of generation gap should be maintained and not overlap at any point in time.

C. Background landscape

The background of it would be landscape, e.g., made up of peak mountain; the base of the mountain would be in green representing the grass, moving upwards it would merge with the color brown showing bare land, then moving upwards to color white representing snow. Figure 3 shows software product line architecture data translated onto the hyperbolic tree with human perception of real world environment kept in mind.

Perception of data is easy if the visualization is inspired from the real world environment and its objects known as "data landscape" in software terminology [4], based on the fact that skills used by human mind in interpreting the real world environment can be used in perceiving the visualization of "data landscape" [4].

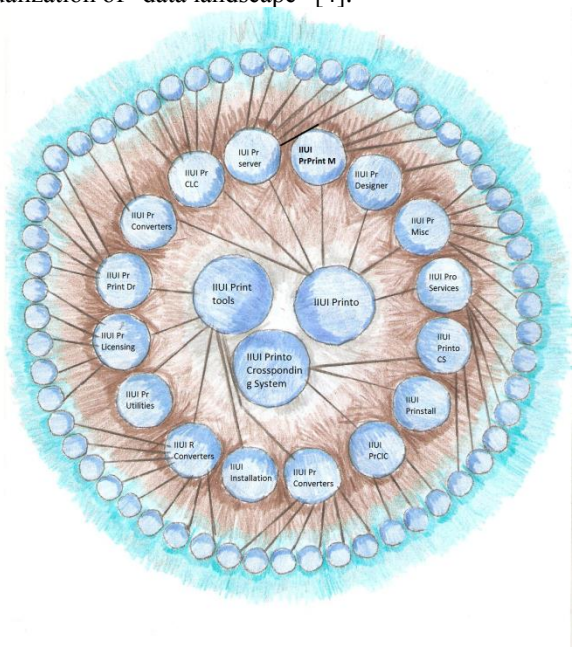

Figure 3. Perception of hyperbolic tree as real world object.

D. Color aid

The concept of "peak mountain" for the background, on which the hyperbolic tree would reside can be achieved with the help of color, as well said by Colin "that color helps in breaking camouflage" otherwise it would be very difficult to determine where or what a certain object is [4]. The use of color is not just about filling an image with color, but one has to bring it as close to real world objects as possible. In fig 3, the circles of nodes are filled with *Lambertian shading*, also the circles shown as objects, are C*asting shadow* on the mountain. Where *Lambertian shading* is known as a method for showing surface shape with the help of shading [4], meaning that if a mixture of color is not used then it is not possible to differentiate between the background and the overlaying objects on them. And C*asting shadows* theory is deduced from the fact that any real world subject can cast shadows either on itself or on the surface it is placed upon. This theory gives us the illusion of perception of height, of an object [4], stating that the specified object is at a height, above the ground that's why it's casting its own shadow on the ground; rather than being at the same level on the ground.

The nodes are also filled with the blue color and the text defining the node is in black color, which brings out the luminance contrast; which states that if the background is low saturation (light color), then the overlaid symbols must be of darker shade [4].

E. *"Affordance"device*

Taking Gibson's *affordance* theory known as perceivable prospective for action [2] into consideration and translating it

to our work, e.g., if the task is to bring second generation of children into focus, it would be highly recommended if "handles" are used [2]. As perceived by Houde, it is rather easy to perceive solution with the help of "handles" than arrows, etc. [1]. Here again, the focus is to bring forth human perception of real life objects, and use those skills as opposed to defining new ones.

IV. CONCLUSION AND FUTURE WORK

This paper starts with identifying the need for information visualization technique for the software product line architecture. It has mentioned the need for not just a good visual structure, but also the need for interaction with it. It further went on to explain the importance of hyperbolic tree and then presented enhancements to the concept of hyperbolic tree introduced by [5] and [6], for the sole purpose of establishing it as a fine means for the representation of software product line architecture data. Along the way, the perception of the human mind was kept in focus based on the rationale that nonfunctional requirement of software product line architecture can only be handled if perception of human mind is focused upon.

There is a need for testing this technique against previously used techniques for representation of software product line architecture. Our future work is based on this.

ACKNOWLEDGMENT

We would like to thank all our teachers and colleagues who helped. A.K, thanks MR. Mushtaq, Ms. Zafar, Mr. Iqbal, Mr. Hussian and Ms. Latif for their endearing support.

REFERENCES

[1] S. Houde. "Iterative design of interface for easy 3-D direct manipulation," Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, Monterey, May, 1992, pp. 135-142. [Retrieved: October, 2013]. doi:10.1145/142750.142772

[2] J. J. Gibson. "The ecological approach to visual perception," Houghton Mifflin, Boston, 1979. (currently published by Lawrence Erlbaum, Hillsdale, NJ.)

[3] S. M. Anstis. "A chart demonstrating variation in acuity with retinal position," Vision Research, vol. 14, no. 7, July, 1974, pp. 589-592. [Retrieved: September, 2013]. doi:10.1016/0042-6989(74)90049-2

[4] C. Ware. "Information visualization:Perception for design," Morgan Kaufman Publishers, 2nd ed, 2004.

[5] J. Lamping and R. Rao. "Hyperbolic Browser:A focus+context Techniques for visualizing large hierarchies," Journal of visual languages and computing, vol. 7, no. 1, March, 1996, pp. 33-55. [Retrieved: September, 2013]. doi: 10.1006/jvlc.1996.0003

[6] J. Lamping, R. Rao, and P.Peter. "A focus+context technique based on hyperbolic geometry for visualizing large hierarchies," In Proceedings of the ACM SIGCHI Conference on Human Factore in Computing Systems(CHI '95), ACM, May, 1995, pp. 401-408. [Retrieved: October, 2013]. doi: 10.1145/223904.223956

[7] B. Johnson and B. Shnedierman. "Tree-maps: a space-filling approach to the visualization of hierarchical information structures," Visualization, 1991. Visualization '91, Proceedings, IEEE Conference on, vol., no., 22-25 October, 1991, pp. 284-291. [Retrieved: September, 2013]. doi: 10.1109/VISUAL.1991.175815

[8] G. G. Robertson, J. D. Mackinlay, and S. K. Card. "Cone trees: Animated 3d visualization of hierarchical information," Proceedings of the ACM SIGCHI Conference on Human factors in computing systems, ACM, 1991 pp. 189-194. [Retrieved: October, 2013]. doi:10.1145/108844.108883

[9] S. Card, J. Mackinlay, and B. Shneiderman. "Readings in Information Visualization - Using Vision to Think," Morgan Kaufmann, 1999.

[10] D. Nestor, L. O'Malley, A. Quigley, E. Sikora, and S. Thiel, "Visualisation of Variability in Software Product Line Engineering," in 1st International Workshop on Variability Modelling of Software Intensive Systems (VaMoS-2007), Limerick, Ireland, 2007. [Retrieved: October, 2013]. doi:10.1.1.136.9399.

[11] S. Ferber, J. Haag, and J. Savolainen. "Feature Interaction and Dependencies: Modeling Features for Reengineering a Legacy Product Line," Software Product Lines (SPLC2): Springer, vol. 2379, August, 2002, pp. 235-256. [Retrieved: September 2013]. doi: 10.1007/3-540-45652-X_15

[12] H. Ye, and H. Liu. "Approach to modelling feature variability and dependencies in software product lines," IEEE, vol. 152, June, 2005, pp. 101-109. [Retrieved: September, 2013]. doi: 10.1049/ip-sen:20045007

[13] S. G. Eick, J. L. Steffen, and E. E. Sumner. "Seesoft-ATool for Visualizing Line Oriented Software Statistics," IEEE Transactions on Software Engineering, vol. 18, no. 11, November, 1992, pp 957-968. [Retrieved: September, 2013]. doi:10.1109/32.177365

[14] A. van Deursen, M. de Jonge, and T. Kuipers. "Feature-Based Product Line Instantiation Using Source-Level Packages," Software Product Lines (SPLC2): Springer, vol. 2379, August, 2002, pp. 217-234. [Retrieved: October, 2013]. doi: 10.1007/3-540-45652-X_14

[15] K. C. Kang et al., "FORM: A feature-oriented reuse method with domain specific reference architectures," Annals of Software Engineering, vol. 5, no. 1, 1998, pp. 143-168. [Retrieved: September, 2013]. doi: 10.1023/A:1018980625587

[16] D. Muthig, and C. Atkinson. "Model-Driven Product Line Architecture," Software Product Lines (SPLC2): Springer, vol. 2379, USA, August, 2002, pp. 110-129. [Retrieved: October, 2013]. doi: 10.1007/3-540-45652-X_8

[17] D. Fey, R. Fajta, and A. Boros. "Feature Modeling: A Meta-Model to Enhance Usability and Usefulness," Software Product Lines (SPLC2): Springer, vol. 2379, USA, August, 2002, pp. 198-216. [Retrieved: October, 2013]. doi: 10.1007/3-540-45652-X_13

[18] S. Salicki, and N. Farcet. "Expression and Usage of the Variability in the Software Product Lines," Software Product-Family Eng (PFE-4): Springer, vol. 2290, Spain, October, 2002, pp. 304-318. [Retrieved: September, 2013]. doi: 10.1007/3-540-47833-7_27

[19] G. Halmans, and K. Pohl. "Communicating the variability of a software-product family to customers," Software and Systems Modeling, vol. 2, no. 1, March, 2003, pp. 15-36. [Retrieved: October, 2013]. doi: 10.1007/s10270-003-0019-9

[20] F. Bachmann et al., "A Meta-model for Representing Variability in Product Family Development," Software Product-Family Eng (PFE-5): Springer, vol. 3014, Italy, November, 2004, pp. 66-80. [Retrieved: October, 2013]. doi: 10.1007/978-3-540-24667-1_6

[21] D. L. Webber, and H. Gomaa. "Modeling variability in software product lines with the variation point model," Sci. Comput. Program., vol. 53, no. 3, December, 2004, pp. 305-331. [Retrieved: September, 2013]. doi: org/10.1016/j.scico.2003.04.004

 ISBN: 978-1-61208-304-9

[22] R. E. Tufte. “Visual explanation:Images and Quantities, Evidence and Narrative,” Cheshire, CT: Graphics Press, 1997.

[23] D. A. Norman. “Things that Make Us Smart,” Reading, MA: Addison-Wesley, 1993

[24] M. Sinnema, S. Deelstra, J. Nijhuis, and J. Bosch. "COVAMOF: A Framework for Modeling Variability in Software Product Families," Software Product Lines (SPLC3): Springer, vol. 3154, USA, August- September, 2004, pp. 197-213. [Retrieved: September, 2013]. doi: 10.1007/978-3-540-28630-1_12

[25] A. V. D. Hoek. “Design-time product line architecture for any-time variability,” Science of Computer Programming, vol. 53, no. 3, Neatherland, December, 2004, pp. 285–304. [Retrieved: October, 2013]. doi:10.1016/j.scico.2003.04.003

[26] K. Abeer, and I. Salma, “Evaluation of Visual structure for Industrial size Software Product Line Architecture,” Proc. of Eighth International Multi-Conference On Computing In The Global Information Technology, Think Mind, France(Nice), July, 2013, pp. 152-157. [Retrieved: October, 2013]. doi:iccgi_2013_7_40_10278