XRaSE: Towards Virtually Tangible Software using Augmented Reality

Rohit Mehra, Vibhu Saujanya Sharma, Vikrant Kaulgud, Sanjay Podder
Accenture Labs, Bangalore, India
{rohit.a.mehra, vibhu.sharma, vikrant.kaulgud, sanjay.podder}@accenture.com

***Abstract*—Software engineering has seen much progress in recent past including introduction of new methodologies, new paradigms for software teams, and from smaller monolithic applications to complex, intricate, and distributed software applications. However, the way we represent, discuss, and collaborate on software applications throughout the software development life cycle is still primarily using the source code, textual representations, or charts on 2D computer screens - the confines of which have long limited how we visualize and comprehend software systems. In this paper, we present *XRaSE*, a novel prototype implementation that leverages augmented reality to visualize a software application as a virtually tangible entity. This immersive approach is aimed at making activities like application comprehension, architecture analysis, knowledge communication, and analysis of a software's dynamic aspects, more intuitive, richer and collaborative. A video demonstration is available at https://youtu.be/QZiWHoFRn4g.**

***Index Terms*—Augmented Reality, Immersive Experience, Software Visualization, Software Comprehension**

I. Introduction

Advances in the nature of software applications in recent times have on one hand tried to take away the complexities inherent in monolithic systems, but at the same time have made applications even more intricate, spanning across multiple technologies and contexts. This, coupled with the fact that numerous interactions between different parts of software make its understanding difficult and error-prone, poses significant challenges in software delivery activities (like software comprehension, knowledge transfer or architecture analysis) among development teams.

Similarly, evolving global software delivery practices increasingly cause team roles and contexts to change pretty quickly. In such scenarios, a single or generic view of the application may not be the best approach suited for everyone. This raises an important question - *Are there new technologies which may be leveraged to rethink how software can be represented in a more intuitive or richer manner?*

Immersive technologies, especially Extended Reality (XR) are interesting candidates for helping represent intricate software applications without the boundaries and limitations of a two-dimensional (2D) projection. XR includes Virtual/Augmented/Mixed Reality approaches (VR/AR/MR), which are witnessing rapid advances in the core technologies as well as devices and maturing ecosystem around them.

Studies have shown that immersion inside a 3D environment provides multiple advantages such as ease of identifying patterns and better recollection compared to 2D equivalents [1], leveraging affordances of natural human perception - spatial memory, navigation and manipulation for better comprehension of 3D visualizations [2] and enhanced creativity [3].

We believe that XR approaches are at a juncture wherein their applicability and use in software engineering, in general, should be explored, especially around areas like representations of various aspects of software systems, wherein 2D representations (and even 3D representation displayed in 2D) are simply an approximation for fitting in the prevalent screen/monitor based displays.

While there have been some approaches that have tried to utilize immersive technologies for software visualization, these have been primarily based on VR. We posit here that AR (which for the sake of simplicity we use to denote mixed and augmented reality approaches) is better suited than VR in this context - due to the non-blocking, augment-able (augmenting the real-world context), natural (e.g. hand gestures) features of AR interactions, which also result in lesser context switching in general. Moreover, because of the open field-of-view in AR, opportunities for fluid collaboration are also inherent.

In this paper, we present *XRaSE* (pronounced as X-rays), a novel prototype implementation that automatically creates an immersive 3D representation of input software code and leverages AR to render/visualize it in the wearer's field-of-view. Our approach makes the software system "virtually tangible" – i.e. it allows her to take advantage of the affordances of natural human perception to interact with it as if interacting with a real-world object. Our approach is aimed at making activities like application comprehension, architecture analysis, knowledge communication, and analysis of a software's dynamic aspects, more intuitive, richer and collaborative. The envisioned users for our setup are project stakeholders like developers, testers, technical architects, project managers, etc.

XRaSE builds upon our previous work [4], which highlights engineering, introspection and intelligence as three important software engineering dimensions that can benefit from using immersive technologies. It is also an extended version of our work on leveraging AR as a common fabric of understanding and insights for global software delivery teams [5].

II. Related Work

Importance of software visualization in 2D space has been actively studied, but in recent years, researchers have begun to explore 3D representations for visualizing software [6]. sv3D [7] is a 3D software visualization framework that builds

978-1-7281-2508-4/19/$31.00 ©2019 IEEE
DOI 10.1109/ASE.2019.00135

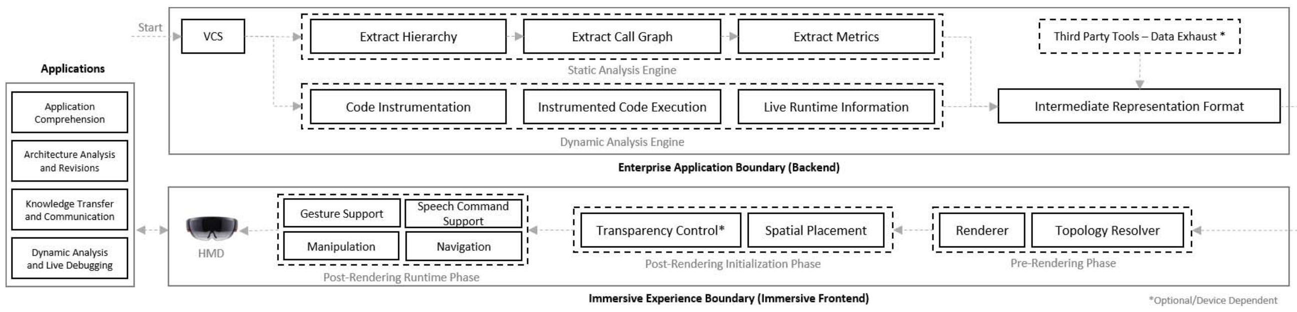
Start
VCS
Extract Hierarchy
Extract Call Graph
Extract Metrics
Static Analysis Engine
Third Party Tools – Data Exhaust *
Code Instrumentation
Instrumented Code Execution
Live Runtime Information
Dynamic Analysis Engine
Intermediate Representation Format
Enterprise Application Boundary (Backend)
Applications
Application Comprehension
Architecture Analysis and Revisions
Knowledge Transfer and Communication
Dynamic Analysis and Live Debugging
HMD
Gesture Support
Speech Command Support
Manipulation
Navigation
Post-Rendering Runtime Phase
Transparency Control*
Spatial Placement
Post-Rendering Initialization Phase
Renderer
Topology Resolver
Pre-Rendering Phase
Immersive Experience Boundary (Immersive Frontend)
*Optional/Device Dependent

Fig. 1. XRaSE System Overview: An AR based immersive application visualization and comprehension framework.

on the 2D SeeSoft pixel metaphor, to visualize, comprehend and interact with large software systems on a 2D display. Similar works include visualizing software systems as cities [8] and forests [9]. Techniques like [10], depicting software production cost-related information on top of modified city metaphor, opens avenues for 3D representation of orthogonal software metrics. While all of these approaches focus on the faster and finer understanding of large software systems in 3D, the constrained navigation problem [11] hinders effective consumption of the content.

To overcome the limitations of 3D software visualization in 2D constrained environment, explorations in XR began. CityVR [12] demonstrates a VR based software gamification tool, that promotes developer engagements in terms of curiosity and interaction time. It takes advantage of the wide range of a user's perceptual senses and their ability to navigate through the visual representation, for better system comprehension. Many similar works representing software systems in VR, again utilize metaphors like city [13] and island [14].

In contrast to the above-mentioned primarily VR-based approaches that switch the user's context from real world to virtual world, restrict effective field-of-view collaboration, and that can involve unnatural affordances of natural human perception, our research aims at leveraging AR aspects of XR, to overcome these limitations.

III. XRaSE

XRaSE focuses on creating an immersive representation of an input software application that can be rendered in AR devices. It aims to facilitate a better comprehension of the application than navigating the source code in the IDE or visualizing on a 2D desktop monitor. Figure 1 shows the schematic view of our approach.

A. Architecture

The source code of the application, fetched from the version control system (VCS), is analyzed to construct the architecture, using standard static analysis tools. These tools allow us to extract the structure and static call graph of the source code. This results in information about the application structure and internal relationships between its constituents (different entities like packages, classes, methods and internal relationships like contains or calls relationship).

We next extract different metrics associated with different elements in the application (e.g. cyclomatic complexity for methods, coupling between objects for classes, etc.) and create a graph-centric model of the entire application (in GraphML or equivalent). This model now contains information about the internal structure, the internal relationships, as well as the metrics associated with individual constituents of the application. We represent it in an intermediate domain-specific representation format because generic GraphML is not well suited to store such multi-layered information.

Apart from the static application structure and metrics, a dynamic analysis engine instruments the source application with custom code snippets that allows the system to capture dynamic system information at runtime (using aspect-oriented programming). This includes information regarding method invocations, data in, data out, performance data, etc. At runtime, this dynamic information is then incorporated in the intermediate representation and sent for further processing on the device itself.

It is important to note that the architecture itself is scalable and can leverage data-exhaust from other heterogeneous tools employed in the delivery environment. For e.g., code quality metrics from Sonarqube, open/closed tickets information regarding a specific class from Jira, etc, can also be overlaid on top of our base representation.

It is important to do further processing on the head-mounted display (HMD) as this is a visualization centric approach and the conditions and environment of the device can help to optimize it. There are three types of processing performed on the device. The *Pre-Rendering* phase is responsible for topology selection and HMD specific rendering of the intermediate representation. Here the topology could be a hierarchical node-link tree, force-directed or simply random (if the number of nodes is very small) representation. Topology selection is dependent upon environmental understanding and complexity/size of the input software.

The *Post-Rendering Initialization* phase is responsible for utilizing the lighting estimation output of the device for the model placement in the real world and deciding its transparency/contrast. This is very important in an AR-based device as for example, very bright lighting may cause semi-transparent parts to be invisible and therefore needs to be

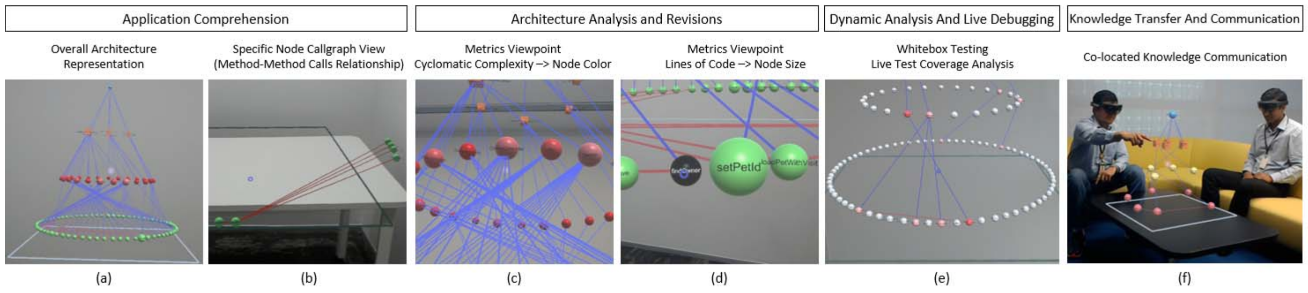
Application Comprehension
Architecture Analysis and Revisions
Dynamic Analysis And Live Debugging
Knowledge Transfer And Communication
Overall Architecture Representation
Specific Node Callgraph View (Method-Method Calls Relationship)
Metrics Viewpoint Cyclomatic Complexity -> Node Color
Metrics Viewpoint Lines of Code -> Node Size
Whitebox Testing Live Test Coverage Analysis
Co-located Knowledge Communication
setPetId
(a)
(b)
(c)
(d)
(e)
(f)

Fig. 2. Immersive software representation using XRaSE. (a-d) Depicting static 3D structure, overlaid metrics, navigation and manipulation viewpoints of open-source Petclinic application in a Microsoft HoloLens HMD. (e) Dynamic test coverage analysis by executing an existing test suite. (f) Collaborative knowledge communication using multiple HMDs.

considered to set the transparency values. Further, the spatial map (mesh-based map of the wearer's surroundings) provides cues to where the model can be placed/anchored in the physical environment. For effective comprehension, the model should ideally be placed on a large flat surface with no physical obstructions on/around the surface, allowing occlusion-free view and navigation.

Finally, our approach needs to allow the user to interact with the application visualization. So, the *Post-Rendering Runtime* phase utilizes HMD specific gestures and voice support (if available) for enabling runtime manipulation and navigation of the rendered application model. This could help users to zoom in or out, change layouts, and do more advanced interactions like filtering only a small part of the overall complex model or type of relationships.

As an outcome of this approach, the user is able to place the generated model in the real-world (as if it was a tangible entity), see and move through and around the application structure, inter-element relationships, metrics and insights in 3D space, as well as is able to interact with (navigate, manipulate) it for specific metric based insights. Multiple viewpoints, topologies and metrics combined with real-life collaborative visualizations enable multitude stakeholders to effectively comprehend, analyze and communicate complex software applications.

B. *Implementation*

To showcase the applicability of the proposed approach, we have implemented *XRaSE*, a novel prototype that can process Java-based applications and provides many of the features we described here, in a Microsoft HoloLens[1] based AR setup. The backend framework is developed using Java (Spring Boot) and Neo4J[2] and employs other third-party open-source tools like QDox Parser[3], Java call-graph[4] and Java CK metrics[5] for static analysis and AspectJ[6] for dynamic analysis. The immersive frontend is developed using Unity3D[7] and Microsoft's Mixed Reality Toolkit[8].

Figure 2 shows the snapshots of the developer's field-of-view while interacting with the software representation generated by processing the open-source PetClinic application[9]. The entire process is completely automated and takes about a couple of minutes to finish processing (approx. 1 minute for every 10K lines of code). *XRaSE* currently supports both static and dynamic (live) analysis and multiple supported insights like frequency hotspots, performance analysis, metrics view, etc. These can be used to enhance multiple business use-cases like comprehension, architecture analysis, knowledge transfer, etc. For e.g., Figure 2f demonstrates two stakeholders collaborating with each other using a common immersive software representation but might have varied and independent role-specific insights/metrics overlaid on top of the base representation, for better understanding. More details can be found in the supplemented demonstration video.

C. *Proposed Evaluation Study Design*

Because of the inherent benefits of immersive representations enabled through AR, as outlined earlier, we posit that typical activities related to software delivery can benefit significantly by utilizing AR. However, we recognize the need for performing in-depth user studies to understand the trade-offs between the rich user experience enabled by AR and the higher, initial interaction, comprehension complexity and are looking forward to evaluating this. The following is the outline of the user study that we plan to conduct in our organization with multiple relevant stakeholders. The study intends to evaluate the comprehension and analysis capabilities of *XRaSE* w.r.t existing 2D approaches and builds upon the guidelines proposed by Merino et al. [15] for improving the quality of software visualization evaluation.

Participants: The user study will be conducted with *n* number of participants divided into four equal-sized groups. Each group will follow a similar composition in terms of participant's role within the organization, where each group

[1]https://www.microsoft.com/en-us/hololens
[2]https://neo4j.com
[3]https://github.com/paul-hammant/qdox
[4]https://github.com/gousiosg/java-callgraph
[5]https://github.com/mauricioaniche/ck
[6]https://www.eclipse.org/aspectj

[7]https://unity.com
[8]https://github.com/microsoft/MixedRealityToolkit-Unity
[9]https://github.com/spring-projects/spring-petclinic

will have a mixture of novice developers, experienced developers and development managers. This will allow us to gain a broader perspective on how our approach fares w.r.t varied experience and possibly varied generations of developers.

Applications: We have chosen two open-source Java-based applications which are comparable in size, complexity and number of nodes (packages + classes + methods) for this study.

Methodology: For this study, we will focus on two mediums of interaction. First, a 2D medium incorporating IDE, static reports and 2D visualizations and second, a 3D AR medium enabled by Microsoft HoloLens which runs *XRaSE*. The participants will be asked to perform a set of t tasks, first on one application using either 2D or 3D medium, and then on the second application using the other medium. Since for two mediums and two applications, the total possible combinations are four, hence the four groups. This methodology is chosen to eliminate both fatigue bias (not able to perform well with the second application due to fatigue caused by performing tasks on first application) and learning bias (being able to use the knowledge gathered by performing tasks in the first medium, during carrying out tasks in the second medium).

Tasks: The tasks are divided into 5 groups - structure, metrics, dependency, code and debugging, and are arranged in increasing order of complexity. Some of these tasks include *locating an interface class within package x*, *finding the best candidate for a god class*, etc. The tasks are based upon the developer's information needs and are designed to replicate typical problems that a developer has to answer multiple times during a work-day [16].

Data Collection: Data collection will be performed in three phases. First, a Pre-Study questionnaire will be filled by the participant, which includes details and metrics like name, age, gender, total development experience, prior exposure to immersive technologies, etc. Second, the moderator will measure the quantitative attributes while the tasks are being performed. This will include parameters like time to task completion, correctness and learning time (participants will be given training on using MS HoloLens and *XRaSE*). Finally, the participant will be asked to fill the Post-Study questionnaire, which will include natural language queries to answer both qualitative and quantitative parameters like cognitive load, engagement, usability and recollection. Finally, we are also exploring to employ biometric scanners to quantitatively analyze the aforementioned qualitative parameters.

IV. Conclusions and Future Directions

The paper presented our ongoing research in exploiting the benefits of AR/MR techniques creating tangible representations of software systems. This approach is aimed at supplanting several software engineering activities, such as application comprehension, architecture analysis, knowledge transfer, and dynamic analysis, among others. As the first step in this research, we have implemented a prototype that uses AR to automatically create a 3D representation of a software system in the wearer's field-of-view. This is then further annotated with various metrics and insights of importance to the wearer. As the approach augments the field-of-view of the wearer, it mitigates many of the problems associated with using more restrictive VR-based approaches in software engineering and keeps the user connected with his real workspace. We are also in the process of initiating a study to evaluate the user experience and feedback on typical software engineering tasks.

We have received encouraging initial feedback including how this triggers fresh discussions around next-gen tools and how this would facilitate comprehension during complex software testing and knowledge transfer activities by distributed teams. We are also engaged in creating specific overlay sets concerning different activities/use cases to make the experience more consistent as well. The user study that we are setting up will help us understand the impact of such a representation for the on-ground delivery teams. This will open further avenues to incorporate user feedback into *XRaSE*.

References

[1] P. Irani and C. Ware, "Diagrams based on structural object perception," in *Proceedings of the Working Conference on Advanced Visual Interfaces*, ser. AVI '00. New York, NY, USA: ACM, 2000, pp. 61–67.

[2] A. Elliott, B. Peiris, and C. Parnin, "Virtual reality in software engineering: Affordances, applications, and challenges," in *IEEE/ACM 37th IEEE International Conference on Software Engineering*, May 2015.

[3] M. Oppezzo and D. L. Schwartz, "Give your ideas some legs: The positive effect of walking on creative thinking." *Journal of experimental psychology: learning, memory, and cognition*, vol. 40, no. 4, 2014.

[4] V. S. Sharma, R. Mehra, V. Kaulgud, and S. Podder, "An immersive future for software engineering: Avenues and approaches," in *Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results*, ser. ICSE-NIER '18. ACM, 2018.

[5] V. S. Sharma, R. Mehra, V. Kaulgud, and S. Podder, "Extended reality in global software delivery: Towards a common fabric of understanding and insights," in *Proceedings of the 14th International Conference on Global Software Engineering*, ser. ICGSE '19, Piscataway, USA, 2019.

[6] A. R. Teyseyre and M. R. Campo, "An overview of 3d software visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 1, pp. 87–105, Jan 2009.

[7] J. I. Maletic, A. Marcus, and L. Feng, "Source viewer 3d (sv3d) - a framework for software visualization," in *25th International Conference on Software Engineering, 2003. Proceedings.*, May 2003, pp. 812–813.

[8] R. Wettel and M. Lanza, "Visualizing software systems as cities," in *2007 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis*, June 2007, pp. 92–99.

[9] U. Erra, G. Scanniello, and N. Capece, "Visualizing the evolution of software systems using the forest metaphor," in *2012 16th International Conference on Information Visualisation*, July 2012, pp. 87–92.

[10] T. Panas, R. Berrigan, and J. Grundy, "A 3d metaphor for software production visualization," in *Proceedings on Seventh International Conference on Information Visualization, 2003. IV 2003.*, July 2003.

[11] A. J. Hanson and E. A. Wernert, "Constrained 3d navigation with 2d controllers," in *Proceedings of the 8th Conference on Visualization '97*, ser. VIS '97. Los Alamitos, CA, USA: IEEE Computer Society Press.

[12] L. Merino, M. Ghafari, C. Anslow, and O. Nierstrasz, "Cityvr: Gameful software visualization," in *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Sep. 2017, pp. 633–637.

[13] J. Vincur, P. Navrat, and I. Polasek, "Vr city: Software analysis in virtual reality environment," in *IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, July 2017.

[14] M. Misiak, D. Seider, S. Zur, and A. Schreiber, "Immersive exploration of osgi-based software systems in virtual reality," in *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, March 2018.

[15] L. Merino, M. Ghafari, C. Anslow, and O. Nierstrasz, "A systematic literature review of software visualization evaluation," *Journal of Systems and Software*, vol. 144, pp. 165 – 180, 2018.

[16] V. S. Sharma, R. Mehra, and V. Kaulgud, "What do developers want?: An advisor approach for developer priorities," in *Proceedings of the 10th International Workshop on Cooperative and Human Aspects of Software Engineering*, ser. CHASE '17, Piscataway, NJ, USA, pp. 78–81.