An Immersive Future for Software Engineering - Avenues and Approaches

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

ABSTRACT

Software systems are increasingly becoming more intricate and complex, necessitating new ways to be able to comprehend and visualize them. At the same time, the nature of software engineering teams itself is changing with people playing more fluid roles often needing seamless and contextual intelligence, for faster and better decisions. Moreover, the next-generation of software engineers will all be *post-millennials*, which may have totally different expectations from their software engineering workplace. Thus, we believe that it is important to have a re-look at the way we traditionally do software engineering and immersive technologies have a huge potential here to help out with such challenges. However, while immersive technologies, devices and platforms, have matured in past few years, there has been very little research on studying how these technologies can influence software engineering. In this paper, we introduce how traditional software engineering can leverage immersive approaches for building, delivering and maintaining next-generation software applications. As part of our initial research, we present an augmented-reality based prototype for project managers, which provides contextual and immersive insights. Finally, we also discuss important research questions that we are investigating further as part of our immersive software engineering research.

CCS CONCEPTS

• **Human-centered computing** → **Mixed / augmented reality**; **Virtual reality**; • **Software and its engineering**; • **Social and professional topics** → *Project and people management*;

KEYWORDS

Software Delivery, Immersive Experience, Augmented Reality, Virtual Reality

1 INTRODUCTION

Software engineering is evolving in many ways. Applications are becoming more intricate [16] (distributed, server-less, reactive). Software development teams are becoming more decentralized [1] wherein there will be a breakdown of traditional software engineering roles [7]. Software maintenance is also becoming more insights-driven where maintenance teams use deep insights from log analysis and forensics to identify deeply embedded root causes and make precise maintenance fixes. These changes bring several challenges for an optimized and industrialized software engineering - How can roles comprehend the complex and interconnected architecture and delivery data of such applications? How do distributed team members share a common context for better collaboration and coordination? How are software analytics insights [20] delivered in a frictionless manner to facilitate optimum delivery management?

At the same point in time, it is natural to expect and see a changing demographic of who the software engineers be - from *Generation X (1965 - 1979)* to *millennials (1980 - 1999)*, to post-millennials (or *Generation Z, 2000 - Date*)[1]. A recent study [19] identifies Generation Z as the most passionate segment when it comes to virtual reality (VR), followed by millenials and Generation X, respectively. Generation Z software engineers might have completely different expectations of how software is designed, developed and maintained. Just like the now entrenched use of consumer technologies like social media, rich collaboration platforms, and cloud-based editors, will the Generation Z cause software engineering practices to adopt immersive technologies in novel ways?

Immersion refers to creating a sense of physical presence in real or imaginary worlds. Studies have shown that immersion inside a 3D environment provides multiple advantages. Some experiments suggest that 3D representations of information structures are easier to identify and recall than 2D equivalents [8] - the conventional mode for traditional software engineering practices. Immersive experiences leverage affordances of natural human perception - spatial memory, motion, manipulation and feedback for better comprehension of 3D visualizations [4] and enhanced creativity [17]. Stereoscopic vision pertaining to the perception of depth can be a great benefit in disambiguating complex abstract representations [12]. It also helps the viewer to judge relative size of objects and distances between them. Also, the intuitiveness of immersive experiences is an incentive to explore how it can augment the intelligence of software engineering actors.

Finally, we are seeing rapid advances in Augmented reality/Mixed reality (AR/MR) devices and rapidly maturing ecosystem in this area [2]. Analysts predict that the business use of VR/AR/MR will overtake personal use imminently [21]. These factors - rapidly evolving nature of software and software delivery, the growing maturity and rapid adoption of immersive technologies, and potentially significant benefits to the future software engineering

[1] https://www.uschamberfoundation.org/reports/millennial-generation-research-review

ICSE-NIER'18, May 27-June 3 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5662-6/18/05…$15.00
https://doi.org/10.1145/3183399.3183414

practices and tools - encourage a rigorous evaluation of immersive technologies in software engineering methods and tools. We posit that introducing Immersive technologies in different areas of software engineering, spanning architecting and creating software systems, comprehending/tracing through complex systems and evolving processes, augmenting intelligence for project roles (management and execution), as well as rich contextual collaboration on different tasks in the software development life cycle (SDLC), will improve current software engineering practice and experience. This is the focus of this paper.

2 RELATED WORK

A good recent starting point for related work could be RIFTSKETCH and IMMERSION [4], proposing a live coding and code review environment, respectively, that takes advantage of affordances provided by virtual reality - spatial cognition, manipulation and navigation to potentially result in higher productivity, lower learning curve and increased developer satisfaction. While this approach presents the benefits of leveraging virtual reality in software engineering, it lacks the necessary motivation and futuristic vision describing what, why and how - diverse categories, roles and phases in traditional software engineering can be benefited by incorporating elements of immersion (not just VR) into day to day activities, which is the focus of this paper.

Much of the work amalgamating immersive technologies with traditional software engineering has been conducted in the field of software visualization for enhanced program and system comprehension. VR City [22] is a software visualization tool that employs a modified city metaphor [3] to analyze complex software systems and corresponding metrics in a virtual reality environment, reducing cognitive load during the maintenance phase. Similar works include representing software systems as cities [11], forests [5], parks [10] and custom visualization languages viz. COOL [13], rendered in CAVE environment. ExplorViz [6] is a web-based live trace visualization tool that leverages WebVR[2] to experience the existing 3D trace model in virtual reality using an Oculus Rift[3] HMD. Custom gestures and best practices for interacting and manipulating the trace model - translation, rotation, zoom, selection and reset are supported using Microsoft Kinect[4]. CityVR [15] demonstrates a VR-based software gamification tool, promoting developer engagements in terms of curiosity, challenge and interaction time.

While all of these approaches focus on a very specific problem of system comprehension during the development and maintenance phases of SDLC - restricted only to developers and virtual reality, our research aims to span across multiple software engineering problems - Engineering, Introspection and Contextual Intelligence, proposing needs and solutions for a diverse set of roles, by utilizing coherent set of immersive technologies including virtual, augmented and mixed reality.

[2]https://www.webvr.info
[3]https://www.oculus.com/rift
[4]https://developer.microsoft.com/en-us/windows/kinect

3 THE IMMERSIVE SOFTWARE ENGINEERING APPROACH

Immersive approaches have a potential to not only enrich existing activities in software engineering, but also enable completely new experiences in and across its different phases. This, however, makes it difficult to coherently place such immersive approaches in the context of software engineering.

While there could be multiple ways to categorize how immersive approaches can impact software engineering, we have chosen a categorization based on the nature of the immersive activities in them. These could either be aimed to help engineer new software systems, or to help augment the intelligence of the actors involved, or to help in understanding and optimizing existing systems. This categorization emerged from discussions and brainstorming sessions with relevant stakeholders within our organization. Figure 1 shows a schematic depicting these categorizations and the sub-areas in these three important dimensions.

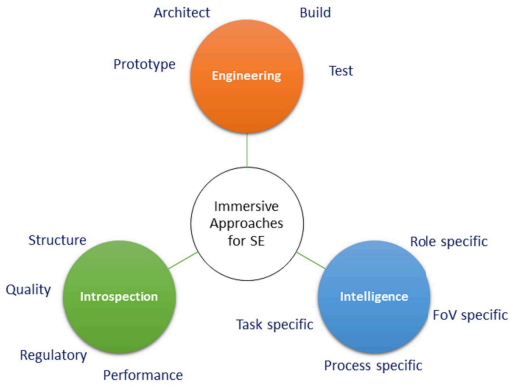

Figure 1: A Broad categorization of Immersive Approaches for Software Engineering

Immersive Engineering: This category pertains to use of immersive approaches in the different activities in SDLC, essentially impacting how we engineer and create software systems. We posit that virtual and augmented reality can become powerful tools in the hands of software project teams, especially to visualize, interact and collaborate on multi-dimensional aspects of the system being built. The premise here is two-fold: firstly, that a software system is not inherently 2-dimensional, and therefore better visualized in an immersive device; and secondly, that we can transform primitives of how we design, build and interact with things in the real (three-dimensional) world, into equivalent actions on virtual entities like software components or their assembly, and therefore make the engineering more intuitive. This, for example, can happen at the point of prototyping and architecting the system, wherein multiple distributed team members can see a complex system's evolving design or architecture in 3D, therefore having a more intuitive basis to do this, and collaboratively bring together and refine different parts of the system. Similar possibilities exist in other phases.

Note, however, that the significance or impact of using immersive approaches across various SDLC activities will vary and may be based on the traditional approach the team is more inclined to

take in absence of such immersive extensions. For example, in the Build phase, a team that is already comfortable composing systems from existing components/services and then mapping them on (say) possible cloud/network topologies, will find using immersive approaches for this much more intuitive, than a team which primarily creates custom code or performs low-level programming.

Immersive Intelligence: This pertains to immersive approaches being applied in an orthogonal manner to the activities being performed in the SDLC, to augment the intelligence of the team members based on their context. Immersive approaches based on augmented or mixed reality can provide a wealth of context on, where the person wearing the device is, what is in the field of view, what is (s)he focusing at, etc. These coupled with the identity/role and the project context of the wearer and in-process project information, can be used to annotate the field of view with important insights and recommendations. For example, one of our own implementations described in the next section, helps a project manager quickly identify the team members' progress and needs for guidance without disturbing her/him. Such intelligence augmentations could also be based upon what phase of the overall process the wearer is in, and the same view can be augmented with different insights or recommendations based upon that. This can thus act as an immersive digital buddy/advisor to the wearer. Our explorations in this category have led us to strongly believe that immersive, context-aware and interactive intelligence augmentation and advice for project actors, can potentially help seamless, faster and informed decisions and actions.

Immersive Introspection: This category pertains to using immersive approaches for software system introspection w.r.t different perspectives like security, compliance, quality, performance or structure, wherein an ability to overlay different aspects/layers of a system as well as its temporal behavior acrosss these aspects can prove to be very useful. The very fact that software systems are increasingly becoming more intricate, spread across multiple platforms with multiple dimensions of concerns/policies, is a clear motivation to be able to project multi-dimensional and multi-aspect views of existing systems from the perspective of maintaining, reflecting upon and improving them. For example, from a data-related compliance perspective, it is important to not only see where data enters or leaves the software system, but also, how did the data travel through the system, the various sub-systems that may have modified it, the data stores that may have persisted it, the physical connections that may have made the data visible, even if intermittently, and finally, how the system itself may have evolved while the data was in it. A simple thought-experiment can motivate that approaches that can help choose and see all these layers, interactions, movements as well as the temporal evolution of the data and the system wherein one can step into the system at a moment of time, can fare much better than a two-dimensional projection and logs on the monitor. Credence to this is lent by studies that humans are really good at detecting visual patterns [14], and hence a rich multi-dimensional field of view can possibly help an expert identify an anomalous pattern in a software system much more intuitively and help optimize that aspect better and faster.

We are witnessing maturity of immersive devices and platforms that are evolving towards being more intuitive, lighter and cheaper, and richer communities of practice are forming around them. We

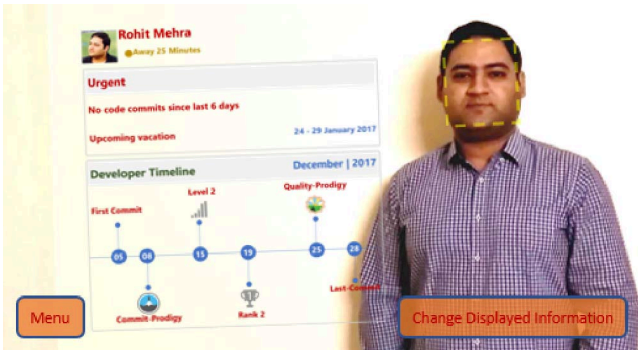

Figure 2: Immersive Project Manager's View - Upon successful facial recognition, manager's field of view displaying contextual knowledge about the subject, including monthly timeline and gamification statistics.

posit that Immersive software engineering will emerge as an important area of research and AR/VR/MR technologies will increasingly be employed to complement, augment and extend the set of activities we see today in the SDLC and beyond. The categorization that we have introduced here can serve as a template to curate and study such rich set of immersive experiences and their implications for software engineering, and we expect this to get extended and refined as the field matures.

4 AN EXAMPLE IMPLEMENTATION: IMMERSIVE PROJECT MANAGEMENT

Some of our initial research is focussed on how immersive experiences can bring in intelligence to software delivery project roles. One such prototype we have implemented is called the Immersive Project Management, which is an early iteration of the thought experiments being conducted to further enrich the Intelligence categorization leveraging augmented reality technology to address the following two problems.

- **Effective Coordination using Contextual Knowledge**: Coordination and communication are critical in software development and team performance enhances if team members have a common comprehension of the task that will be performed and of the involved coordination [9]. This is especially true in case of project managers, facilitating frequent team and developer interactions to continuously guide them and manage the project effectively. Keeping track of every developer in a large-scale team is difficult, hindering effective coordination. For a project manager, answering questions like who is working on what, how is he/she performing - becomes difficult during interaction sessions, not to mention time-consuming.
- **Developer Interruptions**: Contrary to the above mentioned problem, very frequent developer-manager discussions lead to developer interruptions. A programmer takes between 10-15 minutes to start editing code after resuming work from an interruption [18] and is likely to only get just one uninterrupted 2-hour session in a day. As a manager, it is important to reduce the frequency of discussions, while

still being able to have objective information to guide the developer if needed.

Our current prototype can project relevant, in-process notifications and contextual information in manager's field of view, pertaining to a specific developer using a Microsoft HoloLens[5] head-mounted display. The prototype is pre-configured to exhibit a set of insights upon successful developer identification viz. monthly developer timeline, code commit statistics and recently owed technical debt - among others. Current version supports developer identification using facial and marker[6] recognition. Since markers can be tracked from a considerable distance (2-5 meters) with significant accuracy, the augmented manager can analyze the team and developer information without interrupting them, while on a workplace walk (a standard practice in the industry). Facial recognition is used to project contextual information while in close visual range with the developer, to support effective coordination and guidance based on real-time objective data.

Figure 2 shows a developer time-line hologram floating in manager's field of view while interacting with the developer, using facial recognition. Switching between insights is enabled using gaze and airtap gestures supported by HoloLens.

5 FUTURE WORK AND CONCLUSIONS

Continuously evolving size and complexity of modern software applications, the changing dynamics of software project teams, needs for contextual intelligence regarding evolving applications, and rapid changes in the technological landscape, incentivize incorporation of novel and modern technologies. Immersive approaches based on AR/MR/VR technologies are apt candidates, leveraging cost, time and cognitive benefits for various roles involved in a traditional software delivery and maintenance. This paper attempts a vision and provides motivation to weave immersive technologies within software engineering practices in this context. However, this is a nascent area and needs an organized research effort to create the essential foundations. Among the many challenges that the area posses, we believe these are some of the important questions that the community collaboratively needs to study and address:

- Which software engineering activities lend themselves innately to immersive use cases and what are the trade-offs in complexity and impact?
- How do we model elements and interactions of systems, processes and teams as first class entities in the immersive space?
- What are the boundaries of an immersive experience in the application engineering context, before it becomes intrusive?

In this paper, we have ventured to answer aspects of the first of these challenging questions. We have presented three categories of immersive approaches that can bring value to software engineering and we present a few initial examples for each of them. We also presented our prototype implementation for an augmented project manager which provides insights and recommendations based on the context extracted from her/his field of view, while reducing developer interruptions and guiding the manager to those who need his guidance the most. The prototype is being continuously improved and new insights and data sources are being added to it as we are evangelizing this concept and soliciting responses and suggestions from within our organization. Parallely, we are also curating a catalog of more such approaches in different phases of the SDLC and refining our categorization of immersive software engineering approaches.

[5]https://www.microsoft.com/en-us/hololens

[6]https://library.vuforia.com/articles/Training/VuMark

REFERENCES

[1] Andrew Begel, James D. Herbsleb, and Margaret-Anne Storey. 2012. The Future of Collaborative Software Development. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work Companion (CSCW '12)*. ACM, New York, NY, USA, 17–18. https://doi.org/10.1145/2141512.2141522

[2] Mark Billinghurst, Adrian Clark, and Gun Lee. 2015. A Survey of Augmented Reality. *Found. Trends Hum.-Comput. Interact.* 8, 2-3 (March 2015), 73–272. https://doi.org/10.1561/1100000049

[3] Andreas Dieberger and Andrew U Frank. 1998. A city metaphor to support navigation in complex information spaces. *Journal of Visual Languages & Computing* 9, 6 (1998), 597–622.

[4] A. Elliott, B. Peiris, and C. Parnin. 2015. Virtual Reality in Software Engineering: Affordances, Applications, and Challenges. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 2. 547–550. https://doi.org/10.1109/ICSE.2015.191

[5] Ugo Erra and Giuseppe Scanniello. 2012. Towards the Visualization of Software Systems As 3D Forests: The CodeTrees Environment. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing (SAC '12)*. ACM, New York, NY, USA, 981–988. https://doi.org/10.1145/2245276.2245467

[6] F. Fittkau, A. Krause, and W. Hasselbring. 2015. Exploring software cities in virtual reality. In *2015 IEEE 3rd Working Conference on Software Visualization (VISSOFT)*. 130–134. https://doi.org/10.1109/VISSOFT.2015.7332423

[7] R. Hoda, J. Noble, and S. Marshall. 2013. Self-Organizing Roles on Agile Software Development Teams. *IEEE Transactions on Software Engineering* 39, 3 (March 2013), 422–444. https://doi.org/10.1109/TSE.2012.30

[8] Pourang Irani and Colin Ware. 2000. Diagrams Based on Structural Object Perception. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI '00)*. ACM, New York, NY, USA, 61–67. https://doi.org/10.1145/345513.345254

[9] Catholijn Jonker, M van Riemsdijk, and Bas Vermeulen. 2011. Shared mental models. *Coordination, organizations, institutions, and norms in agent systems vi* (2011), 132–151.

[10] Pooya Khaloo, Mehran Maghoumi, II Taranta, David Bettner, Joseph Laviola Jr, et al. 2017. Code Park: A New 3D Code Visualization Tool. *arXiv preprint arXiv:1708.02174* (2017).

[11] Claire Knight and Malcolm Munro. 2000. Virtual but visible software. In *Information Visualization, 2000. Proceedings. IEEE International Conference on*. IEEE, 198–205.

[12] Jonathan I. Maletic, Jason Leigh, and Andrian Marcus. 2001. Visualizing Software in an Immersive Virtual Reality Environment. In *in Proceedings of ICSE'01 Workshop on Software Visualization*. Society Press, 12–13.

[13] Jonathan I Maletic, Jason Leigh, Andrian Marcus, and Greg Dunlap. 2001. Visualizing object-oriented software in virtual reality. In *Program Comprehension, 2001. IWPC 2001. Proceedings. 9th International Workshop on*. IEEE, 26–35.

[14] Mark P. Mattson. 2014. Superior Pattern Processing Is the Essence of the Evolved Human Brain. *Frontiers in Neuroscience* 8 (2014), 265. https://doi.org/10.3389/fnins.2014.00265

[15] Leonel Merino, Mohammad Ghafari, Craig Anslow, and Oscar Nierstrasz. 2017. CityVR: Gameful software visualization. *In Proc. of VISSOFT* (2017).

[16] Sebastian Nanz. 2011. *The future of software engineering*. Springer.

[17] Marily Oppezzo and Daniel L Schwartz. 2014. Give your ideas some legs: The positive effect of walking on creative thinking. *Journal of experimental psychology: learning, memory, and cognition* 40, 4 (2014), 1142.

[18] C. Parnin and S. Rugaber. 2009. Resumption strategies for interrupted programming tasks. In *2009 IEEE 17th International Conference on Program Comprehension*. 80–89. https://doi.org/10.1109/ICPC.2009.5090030

[19] Touchstone Research and Greenlight VR. [n. d.]. The 2015 Virtual Reality Consumer Report. ([n. d.]).

[20] Vibhu Saujanya Sharma, Rohit Mehra, and Vikrant Kaulgud. 2017. What Do Developers Want? An Advisor Approach for Developer Priorities. In *10th IEEE/ACM International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE@ICSE 2017*. 78–81. https://doi.org/10.1109/CHASE.2017.14

[21] Marcus Torchia et al. 2017. Worldwide Semiannual Augmented and Virtual Reality Spending Guide. *IDC Report* (2017).

[22] J. Vincur, P. Navrat, and I. Polasek. 2017. VR City: Software Analysis in Virtual Reality Environment. In *2017 IEEE International Conference on Software Quality, Reliability and Security Companion*. https://doi.org/10.1109/QRS-C.2017.88