Human Factors in Design, Engineering, and Computing, Vol. 238, 2026, 1761–1769
https://doi.org/10.54941/ahfe1008404

Architecture and Early Software Validation for Smart Glasses: A Lean Based Approach

Kinderman Barros, Aécio Jorge Macedo Braga, Marcos Silbermann, and Randerson Queiroz

SIDIA, Brazil

ABSTRACT

The expansion of smart glasses represents the next frontier of wearable computing. The main obstacle for teams developing emerging hardware is the gap between building, testing, and validating a connectivity and interface architecture on a device that is not yet accessible. This challenge is amplified in short development timelines, where final hardware becomes available late. Since mobile and wearable ecosystems operate under volatile requirements and intense time-to-market pressure, deploying adaptable process models is essential to mitigate scheduling risks (Vallon et al., 2015). This work presents a strategy for early technical structuring and validation, mapping the Android Jetpack and Jetpack Compose Glimmer frameworks (Google, 2026) to assisted reality displays, and testing low-latency protocols such as Wi-Fi Aware and Bluetooth between smart glasses and other Android devices. Using the Lean methodology, we mitigated the lack of hardware by using Android smartphones to simulate smart glasses, ensuring software was built under realistic constraints. This iterative process consisted of rapid delivery of Proofs of Concept (PoCs) for evaluation, followed by refactoring to consolidate architectural robustness. Jetpack Compose Glimmer was used to design optimized interfaces, generating PoCs to test connectivity and ensure low-latency data flow. The strategy resulted in a reusable module and a set of libraries, consolidating knowledge about the application lifecycle in Head-Mounted Displays and identifying UX bottlenecks. An intermediate test environment simulated the final user experience before integration. By adopting a hardware-agnostic approach, the code and architecture are ready for immediate integration once Android XR smart glasses are available.

Keywords: Smart glasses, Lean, AndroidXR, Bluetooth, Jetpack glimmer

INTRODUCTION

Smart glasses are a growing market technology emerging as a response to the desire for a more practical and deeper connection with technology in daily life. By seamlessly integrating vision and hearing, these devices allow for instant access to information through a core that combines a mounted display and audio, enabling users to access augmented reality. In most cases, these glasses are enhanced by artificial intelligence that interacts via natural

Received October 16, 2026; Revised November 20, 2026; Accepted December 2, 2026; Available online December 1, 2026

© 2026 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

language and responds to voice and hand commands, leveraging Bluetooth, Wi-Fi, and GPS to improve interaction with the user and the world. Extended reality technology has primarily been developed by combining two ways of interacting with the virtual: augmented reality, where the user sees computer-generated graphics in the real world and mixed reality where the user views the real world and computer-generated virtual elements simultaneously (Shatokhin et al., 2025).

With the arrival of smart glasses the market is already seeing a shift in focus from bulky, immersive headsets toward lightweight smart glasses. Driven by a demand for discreet form factors that fit into social environments and daily life, the industry is moving away from isolated experiences toward wearable hardware practical for long-term use. This surge is heavily powered by the integration of AI providing real-time information and contextual assistance. Beyond than focusing on gaming or niche enterprise applications, this new wave targets general productivity and lifestyle enhancements, fundamentally changing how users interact with digital data in the physical world.

It is within this context, specifically in a development environment, that the need arises to experiment with building new features for these emerging devices. The process demands extensive conceptual study and testing, which makes software development for new technologies resource-intensive and, in some cases, costly for a project. It is therefore essential to adopt agile software development strategies, particularly for Proofs of Concept (PoCs) with the potential to evolve into full-fledged software.

Given the difficulty of accessing this class of new XR hardware, our team proposed a development model that minimizes the issues created by the unavailability of the target device. The proposal was driven by the project's own delivery constraints: facing a fixed delivery deadline that preceded any confirmed date for receiving Android XR devices, needed a way to keep architectural and UX work moving without idle time. The model was built on the premise of adapting Lean Software Development (Poppendieck & Poppendieck, 2003) and PoC-driven, hypothesis-first development (Ries, 2011) to the constraints of this project, rather than adopting a heavier, plan-driven process. Two alternatives were considered and set aside: (1) waiting for physical devkits before beginning implementation, which was discarded because it would have compressed all integration and UX validation work into the tail end of the schedule, and (2) relying exclusively on software-only XR emulators, which was discarded because of the lack of inputs necessary on the software-only emulator. The criterion guiding the final choice was the ability to validate both the connectivity and the interface layers under conditions close enough to production hardware to be trustworthy. Using only devices the team already had access to, paired Android smartphones and one Vuzix Blade 2 (a standalone display-equipped smart glass). This paper presents that methodology and discusses the results achieved through the development of a set of focused features.

The remainder of this paper is organized as follows. The next section characterizes the general challenges of developing for wearable hardware that is not yet available, and contrasts the common industry approaches, waterfall development against physical devkits, software-only VR/XR

emulation, and hardware-in-the-loop proxying (Mihalič, Truntič & Hren, 2022) and then before narrowing to the specific methodology this project adopted. We then describe how Lean Software Development was applied in practice, including the smartphone-based test harness, the weekly PoC cycle, and the resulting communication module. We close with the results of this validation effort and the conclusions drawn for future integration with Android XR hardware.

DEVELOPING FOR NEW TECHNOLOGIES USING CURRENT HARDWARE AND SOFTWARE

From a development perspective, the technology industry is undergoing an accelerated transition toward spatial computing and continuous-use wearables. Software developers face significant growing pains as interfaces shift from flat screens with precise touch input to contextual projections controlled by voice, gesture, and gaze. The difficulty lies in learning these new User Interface (UI) and User Experience (UX) paradigms without the tactile and visual feedback that only final hardware can provide.

Before narrowing to the approach adopted here, it is worth situating it against how the industry typically handles this same gap. A waterfall approach paired with physical devkits waiting for a vendor-supplied development kit before writing integration code front-loads risk into the schedule: any delay in devkit availability, which is common for pre-release new hardware, cascades directly into a delayed release and UX assumptions go untested until very late. Software-only VR/XR emulation typically suffer when simulates rendering and input abstractly, unable to play the physical constraints such as restricted field of view, weight distribution affecting head movement that affect user experience; conclusions drawn from a pure emulator often do not transfer cleanly to the physical device. A third approach, closer to the one adopted in this project, is hardware-in-the-loop (HIL) proxying, long established in automotive and power-electronics engineering as a way to validate control software against a real-time model of hardware that is unavailable, expensive to risk, or still under development (Mihalič, Truntič & Hren, 2022). The strategy presented in this paper adapts that same underlying idea of replacing the unavailable target device with real hardware that behaves similarly enough to expose the same architectural and UX problems to the specific case of Android XR smart glasses, using paired Android smartphones and a standalone headset as the physical proxy instead of a dedicated HIL rig.

A concrete example the team encountered was the loss of Wi-Fi connection itself: on a smartphone, a connection can be used to exchange information between two devices using Wi-Fi Aware communication but some smart glasses have restricted or no equivalent built-in functionality. Without physical hardware to test against, the team had to prototype and iterate on substitute communication patterns, such as Bluetooth connections or standard server-client commnunication, entirely on simulated hardware, with no way to validate how they would actually work in a real implementation until the emulation harness described below was in place.

To foresee these challenges, software engineering for this class of device must adopt extremely decoupled architectures. Because processing Large Language Models (LLMs) directly on a wearable often exceeds battery and thermal dissipation limits, the architectural solution relies on offloading heavy processing to a paired device, such as a smartphone. As Lee et al. (2025) highlight, higher-bandwidth networks can reduce transmission delays to acceptable response times of under one second, which makes validating the latency-versus-reliability trade-off across network bottlenecks a critical step. Within the Android ecosystem for wearables, new technologies such as Jetpack Compose Glimmer (Google, 2026) enable the creation of immersive, adaptive visual elements specifically for AI glasses. Connectivity is managed through protocols such as Wi-Fi Aware, for high-bandwidth peer-to-peer communication, and Bluetooth Low Energy (BLE), with architectures calibrated to switch between the two based on energy and data demands.

METHODOLOGY

Having established the general challenges of developing for hardware that does not yet exist, this section describes in detail how the project addressed them in practice.

Development was grounded in Lean Software Development. Its seven guiding principles, including the elimination of waste, the amplification of learning, and fast, iterative delivery that were formulated by Poppendieck & Poppendieck (2003), adapting lean-manufacturing thinking to software teams. Vallon et al. (2015) demonstrate the industrial viability of these principles specifically in mobile app development under volatile requirements and time-to-market pressure, which is the case that motivated their adoption here. Complementing this, the project's PoC-first posture that treated each iteration as a testable hypothesis about UX or connectivity rather than a committed feature follows the build-measure-learn logic described by Ries (2011) in the Lean Startup approach: a hypothesis is stated, a minimal artifact is built to test it, the result is measured against real (or realistically simulated) usage, and the learning feeds the next cycle. To mitigate schedule risk in this volatile ecosystem, the team adopted short cycles focused on PoCs to validate UX and connectivity assumptions before committing to broader architectural decisions.

Because Android XR smart glasses were not available to the team for most of the project, a controlled simulation environment was built to stand in for the target device. This environment had two components. First, an Android emulator configuration was created to reproduce visual integrity for the PoCs for the "server" and "client" roles without requiring two physical devices. Second, and closer to production conditions, two physical Android smartphone were used as the wearable proxy: the primary phone processed complex requests and transmitted data to the secondary phone, which ran the Jetpack Compose Glimmer interface with its viewport artificially restricted to approximate the narrow field of view (FoV) of assisted-reality displays, and with screen-size limits imposed to emulate the resource constraints expected of the final hardware.

Work was organized around a recurring cycle rather than a single linear pass:

1. Hypothesis. A feature or interaction is framed as a specific, testable assumption (e.g., "a Bluetooth handshake can complete pairing in under X attempts").
2. PoC build. An optimized MVP-style implementation of the smallest slice needed to test the hypothesis is built directly inside the simulated hardware environment described above, rather than in isolation.
3. Simulated validation. The PoC is exercised on the smartphone-based harness, with one developer holding the "client" device and another operating the "server" device, evaluating responsiveness and, where applicable, direct testing on a standalone display-equipped headset for FoV and rendering fidelity.
4. Peer review, then client validation. Results are first validated between developers internally; only after that internal pass does the feature go to the client for validation. Given the experimental nature of the project, review at this stage is limited to basic testing rather than full validation, since complete validation is reserved for final delivery.
5. Refactoring and re-validation. Feedback from both passes is triaged and applied within aproximatly half-week turnaround, producing a new version of the feature, which re-enters the cycle at step 3.

This cycle repeats until every feature in the current requirement scope has been implemented and validated. Once all requirements for a scope are complete, everything built in that cycle is consolidated into a single closed delivery, which undergoes the definitive, final-round validation for that scope before being considered done.

The diagram below details the flow:

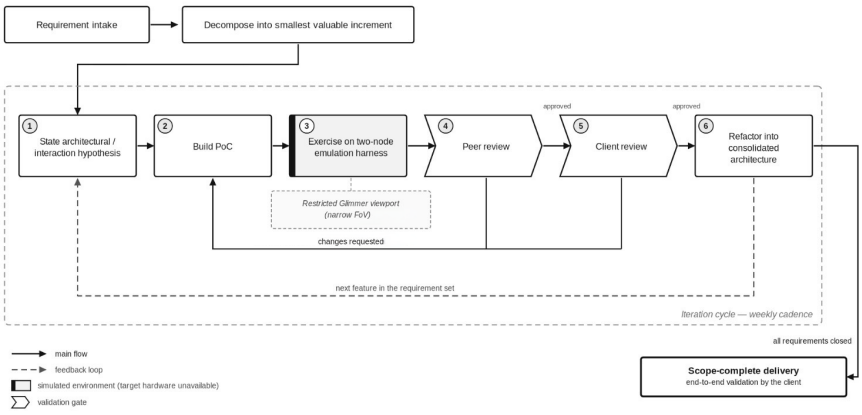


Figure 1: Workflow.

The tools employed during included Android Studio & SDK, Jetpack Compose Glimmer, and an agnostic network module built on the Wi-Fi Aware and Android Bluetooth APIs.

RESULTS

The proposed validation strategy enabled the independent assessment of the system's connectivity and interface layers prior to the arrival of the final Android XR hardware. This decision was made because of a structural constraint common to projects targeting new hardware: the target device, an Android XR-based smart glasses, was not yet commercially available during development, rendering direct testing impossible. To foresee this limitation, the team adopted a proxy hardware strategy relying on three devices: a complementary Android device used as a host service, another Android smartphone as the client, used to prototype and validate inter-device communication, and a standalone Vuzix smart glasses unit as the display and used to receive inputs, used to validate the interface and interaction layer under realistic optical and ergonomic conditions.

This separation was a deliberate architectural decision to treat connectivity and interface as two independently testable concerns, each isolated within its own Android activity, representing a specific device role (host phone, glasses-as-client, standalone glasses). By keeping these layers decoupled, the team was able to identify hardware compatibility issues early, validate key interaction mechanisms in parallel and iteratively refine the implementation during development without instability in one layer masking or delaying the validation of the other.

Device-to-Device Communication: Vuzix Challenges and Limitations

Communication prototypes were developed to validate device discovery, pairing, message exchange, and continuous data transfer between Android devices. Initially, Wi-Fi Neighbor Awareness Networking was investigated as the primary communication technology. Functional prototypes successfully demonstrated discovery and bidirectional communication between Android smartphones, confirming the feasibility of the proposed architecture in standard Android environments. Validation on Vuzix revealed that the device did not support Wi-Fi Aware communication, preventing the proposed solution from operating on the target hardware. This finding led to the abandonment of NAN as the project's communication strategy.

Bluetooth-based communication was then investigated as an alternative. Functional tests exposed an additional platform-specific limitation: the Vuzix operating system only allowed Bluetooth communication with previously paired devices which prevented the originally intended dynamic connection flow, the smart glasses behaved like a conventional smartphone regarding pairing requirements rather than an accessory that connects automatically. Although this restriction made changes to the original communication design necessary, the prototype allowed the team to identify the limitation early and redesign the connection process.

During successive development iterations, improvements were made to the communication flow, including reducing the number of pairing steps, implementing automatic reconnection mechanisms, enhancing message delivery stability and reducing connection failures. Since connectivity was decoupled from the interface layer via independent Android activities representing each

device role, communication issues could be diagnosed without interference from interface behavior, accelerating debugging and validation. This decoupled communication approach between paired devices is consistent with architectures that offload heavy processing to a companion device while keeping transmission latency within acceptable limits (Lee et al., 2025).

Rapid Display Validation Process Using Vuzix

A standalone smart glasses device equipped with an optical display was used to evaluate the interface under simulated field-of-view (FoV) validating the interface under conditions significantly closer to the intended deployment environment than smartphone-based testing alone.

This testing included interface components such as cards, buttons, navigation elements, lists, notifications, overlays, pagers, and scrolls. This revealed several usability issues that were not evident during smartphone development, including interface elements that became too small, insufficient visual contrast against the optical display and colors that were highly saturated.

Rendering experiments showed that conventional 3D models in GLB format appeared visually flat when shown on the Vuzix's display. Replacing these assets with 2D graphics using forced perspective created an illusion of depth that avoided the rendering limitations observed with native 3D models.

Integration of Results and Gains from the Iterative Process

Independent validation of the connectivity and interface layers enabled continuous system refinement during the development process. Compatibility issues were identified, making the need for architectural adjustments to be implemented without delaying other development phases. Customer feedback during this iterative process also led to multiple interface and communication refactorings, reducing implementation effort in later stages and reducing the risk of extensive rework.

The combination of strategies paired with direct interface validation on a standalone device that covered both the system's connectivity and interface layers, despite the use of distinct yet complementary hardware. Segmenting validation in this way also made the weekly iteration cycle described in the Methodology section directly traceable: each hypothesis could be assigned to a specific layer, tested against the corresponding proxy hardware. This kept a half-week feedback turnaround cycle realisticly consistent with the build-measure-learn logic of hypothesis-driven Lean development (Ries, 2011) and the principles of rapid, iterative delivery found in Lean Software Development (Poppendieck & Poppendieck, 2003).

This entire process was supported by the team's adoption of a Lean approach that was structured around the iterative cycle: building requested features, validating them internally, follow up with the client and rebuilding them based on the feedback received. This development flow enabled rapid course corrections, This allowed corrections and changes to be applied specifically and accurately with each feedback, thus avoiding the accumulation of isues that would only be recorded at the end of the delivery, burdening

the entire development process thus reducing the risk of late-stage rework, and ensured that both communication modules and interface components evolved in alignment with actual usage expectations, a result consistent with the industrial viability of these principles in mobile application development characterized by volatile requirements and time-to-market pressure (Vallon et al., 2015).

CONCLUSION

This article presented a strategy for advanced software prototyping for devices not yet on the market, specifically applied to the emulation of Android XR-based smart glasses. By combining Android smartphones acting as communication proxies with a standalone headset used for direct interface validation, the team was able to test both the system's connectivity and interface layers well before the final hardware became available. Communication prototypes built on Wi-Fi Aware and Bluetooth distributed across separate activities to represent the roles of each future device that made it possible to validate pairing, messaging, and data transfer under realistic conditions. Simultaneously, direct testing on a standalone device with a display allowed for the validation of Jetpack Compose Glimmer (Google, 2026) components against actual hardware.

The Role of the Iterative Lean Cycle

This hybrid approach was only possible thanks to the adoption of an iterative Lean cycle repeated continuously for each new feature or adjustment. This method aligns with the build-measure-learn cycle described by Ries (2011): each testable hypothesis leads to a minimal production, the result of which is measured and feeds back into the next cycle, reducing effort while validating learning. This method transformed the fragmentation of available hardware from a limitation into a controlled testing asset, allowing the team to isolate and validate different system aspects independently while simultaneously converging on a coherent final experience.

Contributions and Resulting Assets

The outcome was a reusable communication module and a set of validated interface components that consolidate the team's understanding of the smart glasses application lifecycle and allowing low-risk integration as soon as Android XR smart glasses become commercially available.

ACKNOWLEDGMENT

This paper was presented as part of the results of the project "SIDIA-M_VST_PLATFORM_AND_APPLICATIONS", carried out by the Institute of Science and Technology – SIDIA, in partnership with Samsung Eletrônica da Amazônia LTDA, in accordance with the Information Technology Law n.8387/91 and article ate the.39 of Decree 10,521/2020.

REFERENCES

Google. Build UI for AI glasses with Jetpack Compose Glimmer. Jetpack Compose Glimmer, 2026. Available in: https://developer.android.com/develop/xr/jetpack-xr-sdk/jetpack-compose-glimmer. Last Acessed: 18/04/2026.

Lee, Zhe-Yu; Yuan-Chun Sun; Yee-Nam Wong; Cheng-Hsin Hsu. Exploring LLM-Based Assistants with Smart Glasses for the Visually Impaired, SEC '25: Tenth ACM/IEEE Symposium on Edge Computing, 1–8. 2025.

Mihalič, F.; Truntič, M.; Hren, A. Hardware-in-the-Loop Simulations: A Historical Overview of Engineering Challenges. Electronics, v. 11, n. 15, p. 2462, 2022. https://doi.org/10.3390/electronics11152462

Poppendieck, M.; Poppendieck, T. Lean Software Development: An Agile Toolkit. Boston: Addison-Wesley Professional, 2003.

Ries, E. The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. New York: Crown Business, 2011.

Shatokhin, O.; Dzedzickis, A.; Pečiulienė, M.; Bučinskas, V. Extended Reality: Types and Applications. Applied Sciences, v. 15, n. 6, p. 3282, 2025. https://doi.org/10.3390/app15063282

Vallon, R. et al. An Agile and Lean Process Model for Mobile App Development: Case Study into Austrian Industry. Journal of Software, v. 10, n. 11, pp. 1245–1264, 2015.