Reflection on Requirements Elicitation and Campus Parking Artifact

Michael Ellis

9/21/2025

Reflection

The requirements elicitation activity and the subsequent development of the campus parking artifact provided a valuable opportunity to practice translating user frustrations into a tangible design solution. My partner interviews revealed how something seemingly mundane—finding parking—quickly escalates into one of the most stressful parts of a student's daily routine (especially at Clemson). Students described arriving on campus early only to circle lots for ten to fifteen minutes, which often led to lateness, reduced focus in class, and general anxiety about commuting. These conversations highlighted that parking was not just an issue of convenience but one of efficiency, fairness, and mental well-being.

Initially, I assumed that the primary solution students wanted was simply "more spaces." However, as I am a student myself, I have no authority on making parking spaces for the University. Students acknowledged that the university could not create new lots overnight, but what they desired was a way to make more informed decisions about where to park before wasting time circling. This realization shifted my perspective: the problem was not a shortage of parking per se, but a shortage of transparency about where parking was currently available. This reframing of the problem directly shaped the design of my artifact.

The process of requirements elicitation also changed my understanding of how to ask the right questions. When I first began, I tended to ask broad or leading questions such as, "Would a parking app help you?" This often produced predictable yes/no answers. As

I gained more experience, I shifted toward scenario-based prompts like, "Describe what happens when you arrive on campus at 9 a.m." or "What do you do if your usual lot is full?" These open-ended questions encouraged my partner to provide richer details about their frustrations and coping strategies. From these responses, I was able to extract more concrete requirements, such as the need for real-time data, role-based filters (student vs. faculty vs. visitor), and simple navigation links for directions to an alternate lot.

One of the most important lessons I learned is that requirements elicitation is not a one-time event but an evolving dialogue. My early assumptions about what features mattered most were reshaped by repeated clarifications. For example, I initially believed detailed statistics and predictive analytics would be highly valuable. In practice, students cared less about projections and more about a quick, color-coded indicator of whether a lot was usable right now. This taught me that sophistication does not always equal usefulness, and that user priorities should outweigh designer preferences.

Connecting this exercise to our semester-long software engineering project, I can see how requirements elicitation functions as the foundation for everything that follows. If the requirements are vague, misaligned, or based on assumptions, the resulting system risks being elegant but irrelevant. Our project teams will need to remember that requirements are best expressed in terms of user actions and outcomes: for example, "A student should be able to identify an available lot within two taps" or "The system should display directions in less than five seconds." These testable requirements not only improve design clarity but also make it possible to evaluate whether the system genuinely solves the identified problem.

In a real-world context, requirements gathering often suffers from similar pitfalls: stakeholders may have conflicting needs, initial interviews may be incomplete, and assumptions may creep into the design process. The parking project underscored the value of iterative communication and the importance of verifying assumptions against actual user feedback. It also illustrated how small design decisions—such as the choice to include a "Notify Me" toggle for nearby lots—can directly address anxieties voiced during interviews. These seemingly minor features become critical once viewed through the lens of user experience.

Ultimately, the reflection component reinforced that software engineering is not just about coding features but about building empathy for the people who will use those features.

Requirements elicitation is the bridge between abstract technical possibilities and the messy realities of human behavior. By practicing these skills in a focused exercise, I gained a clearer sense of how to apply them to larger, more complex projects. I now feel better prepared to enter group discussions with the mindset that every assumption should be tested, every design choice should trace back to a requirement, and every requirement should emerge from real user needs.