Erfahrungsbericht: Entwicklung „CarrierTracking“

Team-Mitglieder: Felix Filser, Florian Kovacsik, Moritz Nentwig, Niko Burkert, Sebastian Hartmann, Viktor Dötzel

Das Projekt begann mit der Vorstellung der verschiedenen möglichen Projekte von EvoBus. Vor allem da schon Mitglieder aus unserer Gruppe eine grobe Vorstellung von der Idee hatten, haben wir das Projekt zur automatischen Erkennung von Ladungsträgern im Raum als unseren Favoriten ausgewählt. Als wir Bescheid bekamen, dass wir an diesem Projekt arbeiten dürfen, haben wir uns zu einer Besprechung getroffen um die Teamrollen auszuwählen. Die Auswahl des Scrum-Masters und der Product-Owner (wir haben zwei Product-Owner ausgewählt, da wir eine größere Gruppe sind) haben wir gemeinsam beschlossen und war schnell geregelt.

Anschließend haben wir unser erstes Scrum-Meeting abgehalten, Ideen für das Produkt gesammelt und unser Product-Backlog erstellt. Das war ein wichtiger Schritt, um Ungereimtheiten und Missverständnisse aus dem Weg zu schaffen. Des Weiteren haben wir grobe Aufgabenthemen auf die Entwickler aufgeteilt. Ich habe mich vorerst dazu entschieden, mich um die QrCode Erkennung zu kümmern. Im ersten Sprint war das Ziel eine passende Bibliothek dafür zu finden und diese grundlegend zu implementieren. Dabei war die erste Hürde eine C#-Bibliothek zu finden, welche gut funktioniert, ausreichend dokumentiert ist, jedoch kostenlos angeboten wird für kommerzielle Zwecke. Für viele Lösungen stehen leider hohe einmalige oder regelmäßige Kosten zu Buche. Mit „ZXing“ habe ich mich schlussendlich für eine Bibliothek entschieden, welche aktuellen Support bietet und einen guten Ruf in der Community besitzt. Bei der Implementierung war außerdem entscheidend, einen Algorithmus zu erstellen, dass ein Bild in kleinere Blöcke zugeschnitten werden kann und diese einzeln auslesen zu lassen, da somit die Bibliothek genauer arbeitet und so dem Problem entgegengewirkt wird, dass „ZXing“ nur maximal 6 QrCodes pro Bild erkennt. Wichtig war außerdem darauf zu achten, dass beim aufteilen der Bilder eine Überlappung stattfindet, da sonst möglicherweise QrCodes auf dem Bild auseinandergeschnitten werden und schlussendlich auch nicht erkannt werden können. Um die Implementierung in Unity zu nutzen, habe ich danach das Script dem Unity-Projekt hinzugefügt. Gleichzeitig musste auch die ZXing-Dll und weitere Dlls aus dem .Net-Framework kopiert werden, was zum Glück gut funktioniert hat.
Ein weiterer großer Punkt war der regelmäßige Aufruf der Bilderkennung im Zusammenhang mit den anderen Codemodulen. Zu diesem Zeitpunkt haben die meisten Teammitglieder ihre Aufgabengebiete einzeln umgesetzt, jedoch fehlte der Zusammenbau aller Teile in ein ganzes Softwarekonstrukt. Dabei fiel auf wie schwierig es ist das Arbeitstempo jedes Teammitglieds miteinander in Einklang zu bringen, da man oft auf Code eines anderen Teammitglieds angewiesen ist, um selbst weiter zu arbeiten.
Das Auslesen der QrCodes und das Zeichnen der dazu gehörigen Ladungsträger wurde auf zwei regelmäßig aufgerufene Funktions-Kreisläufe aufgeteilt. Der eine kümmert sich darum, die Ordner mit den Kamerabildern zu überwachen und bei Ankunft neuer Bilder die Positionen der QrCodes auszulesen und diese Informationen während der Laufzeit abzuspeichern. Der andere Kreislauf kümmert sich darum, regelmäßig die abgespeicherten Positionsdaten zu verwenden, um auf vorher erstellten Stationen Blöcke zu zeichnen, welche die Positionen der Ladungsträger in der realen Welt darstellen sollen.
In den letzten 10 Tagen des Projekts ging es darum, den Teammitgliedern Arbeit abzunehmen und zu helfen, welche noch viel zu erledigen hatten. Dies wurde unter anderem mithilfe von abendlichen Webex-Meetings umgesetzt, welche am Ende in häufiger Regelmäßigkeit vorkamen.
Ein größeres Problem während des Projekts war unter anderem die Erkennung der QrCodes auf Bildern aus weiterer Entfernung. Wir haben mit vielen Testbildern getestet, jedoch war es schwierig

den Verantwortlichen von EvoBus eine verbindliche Aussage zu geben, bis zu welcher Entfernung und Größe der QrCodes das Auslesen zuverlässig funktioniert. Dies hängt stark von der eingesetzten Kamera und den vorherrschenden Lichtverhältnissen ab. Ein weiteres Problem war der Umgang mit Git im Zusammenspiel mit Unity. Meist lief es reibungslos ab, da jeder vor allem an den „eigenen“ Dateien gearbeitet hat, jedoch musste auch jeder auf die gleichen Unity-Scenes zugreifen, was oft zu Merge-Konflikten geführt hat, welche teilweise zeitintensiv wieder gelöst werden mussten.

Neben den technischen Aspekten und der Arbeit mit Unity, konnte ich auch neue wertvolle Erfahrungen im Umgang mit Softwareprojekten im Team gewinnen. Vor allem da jedes Teammitglied eine andere Herangehensweise und einen anderen Programmierstil besitzt, ist es umso wichtiger, das Projekt ausführlich richtig zu planen und Missverständnisse frühzeitig auszuräumen.