Kollaborative Software-Visualisierung für verteilte Entwicklungs-Teams mit SEE (Demo)

Rainer Koschke, *AG Softwaretechnik*, *Universität Bremen*,
Dennis Küster, *Cognitive Systems Lab*, *Universität Bremen*

I. Einführung

In diesem Beitrag einer Tool-Demo zeigen wir den aktuellen Stand unserer Visualisierungsplattform SEE (für *Software Engineering Experience*), an der wir seit vielen Jahren entwickeln. SEE visualisiert Software als Code-City in 3D. Der Fokus hierbei liegt auf der Unterstützung verteilt arbeitender Teams. SEE schafft virtuelle Räume, in denen sich Entwicklerinnen und Entwickler treffen können, um sich über ihre Software auszutauschen. Die virtuellen Räume können mit normalen Desktop-Computern mit Monitor, Tastatur und Maus, aber auch mit moderner VR-Hardware betreten werden.

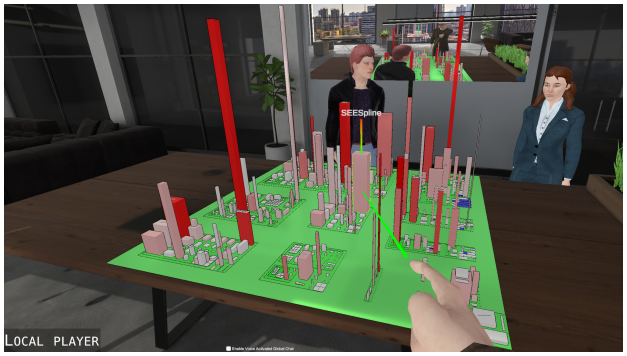

Abb. 1. Virtueller Raum in SEE. Gezeigt wird eine Code-City für SEE selbst.

II. Der Kontext

Der Trend zur verteilten Entwicklung, der längst schon vor der Pandemie eingesetzt hatte, wird weiter anhalten. Große Organisationen sind meist über verschiedene Standorte verteilt. Zudem arbeiten viele Entwicklerinnen und Entwickler gerne im Home-Office, nicht nur wegen pandemiebedingter Einschränkungen, sondern auch weil damit lange tägliche Anfahrtswege wegfallen und sich die Arbeit flexibler mit familiären Aufgaben verbinden lässt.

III. Das Problem

Verteilte Arbeit hat aber den großen Nachteil, dass man anderen während der Entwicklung nicht eben einmal helfend über die Schulter schauen kann, wenn weitere Informationen oder Feedback benötigt werden. Video-Konferenzsysteme und Chats, die eingesetzt werden, um räumliche Distanzen zu überbrücken, helfen hier nur bedingt weiter. Das wissen alle, die es selbst einmal erlebt haben. Zwar kann man den Bildschirm teilen, aber immer nur eine Person hat die volle Kontrolle über die Anwendung und alle anderen müssen diese quasi dirigieren. Zwar gibt es seit einiger Zeit auch IDEs, wie die IntelliJ-basierten IDEs der Firma *JetBrains* mit der Erweiterung *Code With Me*, die kollaboratives Editieren ermöglichen, aber das unterstützt die direkte Zusammenarbeit nur auf der Ebene des oft zu detaillierten Quelltexts und zeigt auch nur, was gerade ist, aber nicht, was früher einmal war. Der Blick für das große Ganze geht hier schnell verloren. Was spielt denn dieser Quellcode für eine Rolle in der Software-Architektur? Welche Auswirkungen auf andere Teile des Systems hat es, wenn wir ihn ändern? Wann und warum wurde er zuletzt verändert und wie hat er sich über die Zeit entwickelt? All das sagt einem der Quelltext alleine nicht.

IV. Die Ziele

Wir wollen Abhilfe bei bekannten Problemen in der verteilten Software-Entwicklung schaffen. In unserem Projekt SEE untersuchen wir die Frage, wie kooperatives Programmverstehen in verteilten Teams in der Software-Entwicklung mit Hilfe von modernen konvergenten Visualisierungstechnologien besser unterstützt werden kann. Unter technologischer Konvergenz verstehen wir an dieser Stelle die enge Integration zuvor unzusammenhängender Technologien. Konkret integrieren wir sowohl herkömmliche Desktop-Hardware (Computer, Tastatur und Maus) und Tablet-Geräte als auch fortschrittlichere Hardware für virtuelle (VR) und erweiterte (AR, engl. *augmented*) Realität so, dass Software-Entwicklerinnen und -Entwickler ein zweckdienliches, einheitliches und zutreffendes Bild ihrer Software über räumliche Distanzen hinweg bekommen. Es wird ein gemeinsamer virtueller Raum erschaffen, in dem sich die Beteiligten in exemplarischen Anwendungsszenarien in der Weiterentwicklung von Software verständigen können. Diesen virtuellen Raum können die Beteiligten mit Geräten ihrer Wahl (Desktop, Tablet, VR oder AR) „betreten“. Darin können Entwicklerinnen und Entwickler nicht nur einen gemeinsamen Blick auf ihre Software, die als Code-City dargestellt ist, und auch die Quelltexte werfen, sondern es werden auch Visualisierungen des Aufbaus und der Abhängigkeiten eines Programms, eingesammelter Laufzeitdaten sowie historischer Änderungen an der Software angeboten. Die Visualisierung bietet eine abstraktere Grundlage, wenn das große Ganze für das Verständnis notwendig ist. Bei den exemplarischen

Anwendungsszenarien streben wir danach, ein großes Spektrum realistischer Anwendungsfälle in der Weiterentwicklung von Software abzudecken. Hierzu gehören das Debugging, die Performance-Analyse, die Analyse der inneren Qualität, das Nachvollziehen früherer Entwicklungen und die (semi)automatische Prüfung der Software-Architektur. Im Projekt arbeiten wir auch eng mit software-entwickelnden Organisationen zusammen, mit Hilfe derer wir partizipativ und iterativ konkrete Anforderungen erheben und deren Umsetzung kontinuierlich entlang der Ideen der Aktionsforschung evaluieren.

V. Die Umsetzung

SEE basiert auf der Spiele-Engine Unity 3D und ist in C# implementiert. Unsere Implementierung ist unter der MIT-Lizenz öffentlich verfügbar[1]. Für Unity haben wir uns entschieden, weil es viele Funktionalitäten mitbringt, die wir benötigen. Wir hatten anfangs die Unreal Engine verwendet, die es jedoch erforderlich macht, Code in C++ zu entwickeln oder aber so genannte Blueprints zu verwenden. Letztere sind eine graphische Notation, die jedoch schnell zu Merge-Konflikten bei textbasierten Versionskontrollsystemen wie Git führt. Neben der leichteren Zugänglichkeit von C# für unsere Studierenden ist auch die höhere Verfügbarkeit von Dokumentation und anderen Formen von Hilfestellungen (wie YouTube-Videos und Entwicklerforen) ein Vorteil von Unity.

Als Visualisierungsform haben wir die Metapher der Code-Cities ausgewählt, weil sie sich für verschiedene Zwecke gut eignet und eine intuitive höhere Abstraktion bietet. Auch viele andere Forschungsgruppen im deutschsprachigen Raum verwenden Code-Cities.

Die Teilnehmenden im virtuellen Raum werden mit menschlichen Avataren dargestellt und sind somit Teil der Szene. Dadurch können wir neben unserem Voice-Chat auch non-verbale Kommunikation in Form von Gesten übermitteln. Zum Beispiel ist es möglich, mit Hilfe eines Lichtstrahls auf Dinge zu zeigen. Diese Geste wird auf alle verbundene Instanzen von SEE übertragen und ist somit von allen von Ferne zugeschalteten Teilnehmenden nachvollziehbar. Wird VR-Hardware benutzt, können wir die Position der VR-Controller, die von den Teilnehmenden in den Händen gehalten werden, nachverfolgen und auf die Handpositionen der Avatare übertragen. Durch inverse Kinematik werden die Körperhaltungen der Avatare so angepasst, dass daraus eine stimmige Bewegung wird.

Gegenwärtig arbeiten wir daran, auch die Mimik zu übertragen. Im Falle von Desktop-Computern mit einer Web-Cam erkennen wir das Gesicht der Person und übertragen dies vor dem Kopf des Avatars, der diese Person in der Szene repräsentiert. Im Falle von VR-Hardware, bei der das Gesicht der Person durch die VR-Brille bedeckt ist, verwenden wir den Facial Tracker von Vive, der die untere Partie des Gesichts sehr zuverlässig erkennt. Damit ist es möglich, Lachen, Sprechbewegungen und auch das Herausstrecken der Zunge zu übertragen. Letzteres ist in formalen Meetings eher selten anzutreffen, wird aber gerne ausprobiert, wenn unsere Probanden das Gerät zum ersten Mal nutzen. In einem Forschungsprojekt mit dem *Cognitive Systems Lab* der Universität Bremen, das sich auf Biosignale sowie die Erkennung und Interpretation von Sprache, Muskel- und Hirnaktivitäten spezialisiert hat, verwenden wir EMG-Sensoren, die auf dem Gesicht unter der Brille angebracht sind, um die Muskelaktivitäten zu identifizieren. Aus diesen Signalen schließen wir auf die Mimik, um sie zutreffend im Gesicht der Avatare abzubilden.

Wir haben verschiedene Methoden implementiert, um Daten über die Software, die es zu visualisieren gilt, zu generieren. Zum einen haben wir eine Anbindung an die Analyseergebnisse geschaffen, die mittels statischer Code-Analyse von der Axivion Suite[2] gewonnen werden. Das hierbei ausgewählte Austauschformat ist GXL. Wir sind auch in der Lage, beliebige Graphen in GXL zu importieren, die von anderen GXL-fähigen Tools generiert werden. Zudem arbeiten wir an Anbindungen an IDEs mit Hilfe des *Language Server Protocol*[3] und an Debugger mittels des *Debug Adapter Protocol*[4] sowie an das Versionskontrollsystem Git. Auf diese Weise werden wir in der Lage sein, sowohl statische als auch dynamische Daten für eine Vielzahl von Programmiersprachen zu gewinnen und diese mit historischen Daten aus einem Versionskontrollsystem anzureichern.

VI. Zusammenfassung

SEE visualisiert Software auf Basis der Code-City-Metapher in virtuellen Räumen, die von Teilnehmenden von verschiedenen Arten von Endgeräten von jedem beliebigen Punkt des Internets betreten werden können. Die Teilnehmenden können unabhängig voneinander mit der dargestellten Software interagieren und jeweils ihre eigene Perspektive einnehmen. Sie selbst werden als Avatare repräsentiert und können sowohl verbal als auch zu einem großen Grad non-verbal miteinander kommunizieren. SEE soll auf diese Weise räumliche Distanzen in verteilt arbeitenden Teams überwinden, die bis dato von existierenden Videokonferenzsystemen sehr eingeschränkt werden.

Wir entwickeln und evaluieren SEE im Kontext eines von der DFG geförderten Forschungsprojekts mit Anwendungspartnern aus der Industrie weiter.

[1] https://github.com/uni-bremen-agst/SEE

[2] https://www.axivion.com

[3] https://microsoft.github.io/language-server-protocol/

[4] https://microsoft.github.io/debug-adapter-protocol/