Industrielle Gemeinschaftsforschung

Schlussbericht vom 31.07.2022

zu IGF-Vorhaben Nr. 20961 BR

Thema

Indoor Positioning Software Framework for Intralogisitics Applikations

Berichtszeitraum

01.01.2020 - 31.07.2022

Forschungsvereinigung

Forschungsgemeinschaft Intralogistik/Fördertechnik und Logistiksysteme e.V.

Forschungseinrichtung(en)

Technische Universität Dresden
Institut für Technische Logistik und Arbeitssysteme
Professur für Technische Logistik (TL)
01062 Dresden

Technische Universität Dresden
Institut für Software- und Multimediatechnik
Professur für Softwaretechnologie (SWT)
01062 Dresden

Forschungsnetzwerk Mittelstand

Gefördert durch:

Bundesministerium für Wirtschaft und Klimaschutz

aufgrund eines Beschlusses des Deutschen Bundestages

Einleitung

In diesem Bericht wird das im Forschungsprojekt IPos Software Framework: *Indoor Positioning Software Framework for Intralogistics Applications* (IPos) erreichte Forschungsergebnis dokumentiert, zusammenfassend dargestellt und bewertet. Auf die in den einzelnen Arbeitspaketen durchgeführten Arbeiten wird eingegangen. Zu Projektbeginn festgesetzte Ziele werden vorgestellt und dem erreichten Ergebnis gegenübergestellt. In diesem Abschnitt wird der dem Projekt zugrunde liegende Problembereich beschrieben. Weiterhin werden einige Hintergrundinformationen gegeben, die für das Verständnis des übrigen Textes nützlich sind.

Projektinhalt

Das IPS-Projekt ist der wissenschaftlichen Weiterentwicklung des technischen Ansatzes für den Entwurf von Indoor-Positionssystemen (IPS) gewidmet zur Aufhebung der strukturellen Inflexibilität dieser Systeme. Es wird der Ansatz der Framework-basierten Entwicklung eines Indoor-Positionssystems untersucht. Dazu wurde im Rahmen des Projektes ein prototypisches Framework für die IPS-Entwicklung und in mehreren Fallstudien evaluiert. Die Praxisrelevanz der erreichten Ergebnisse wurde durch die diesbezügliche Abstimmung mit potenziellen Anwendern aus der Praxis sichergestellt. Diese potenziellen Anwender bilden im Rahmen des Projektes den sogenannten projektbegleitenden Ausschuss. Zur Abstimmung mit den potenziellen Anwendern wurden mehrere Sitzungen des projektbegleitenden Ausschusses sowie eine Umfrage unter den Mitgliedern dieses Ausschusses durchgeführt.

Hintergrund

Im Folgenden geben wir zur Vorbereitung des Abschnitts zur Problemstellung einige Hintergrundinformationen. Indoor-Positionssysteme verdanken ihre hohe Bedeutung der technischen Beschränkungen von Outdoor-Navigationssystemen wie dem Satelliten-gestützten GPS:

Eine korrekte Positionsberechnung kann nur bei Bestehen einer direkten Sichtverbindung zwischen Satellit und Signalempfänger erfolgen

Innerhalb von Gebäuden ist der direkte Signalempfang durch die Gebäudestrukturen gestört. Die Berechnung kann dann aufgrund der Schwäche des auf direktem Wege empfangenen Signales entweder nicht erfolgen, oder sie kann nur unter Inkaufnahme einer hohen Ungenauigkeit erfolgen. Dies ist dann der Fall, wenn die direkte Sichtverbindung zwar gestört ist, aber das Satellitensignal dennoch über eine indirekte Sichtverbindung, also etwa durch Reflexion und Fenstereintritt in das Gebäude, empfangen werden kann. Durch die unvorhersehbare Laufzeitverlängerung des Satellitensignales kommt es aber bei der Positionsbestimmung auf der Grundlage eines indirekt empfangenen Signales zu mehr- oder weniger hohen Ungenauigkeiten. Diese Ungenauigkeiten sind für viele Anwendungen, die eine Positionsbestimmung innerhalb von Gebäuden erfordern, nicht akzeptabel. Aus diesem Grunde wurden spezielle technische Lösungen entwickelt, die eine genaue Positionsbestimmung auch innerhalb von Gebäuden ermöglichen.

Relevante Technologien

Solche Indoor-Positionssensortechnologien existieren heute in großer Zahl. Die Arbeit (Brena [et al.] 2017) gibt einen guten Überblick über die vorhandenen Indoor-Positionssensortechnologien. Für das IPos-Forschungsprojekt sind insbesondere die Technologien *Beacon-basierte Positionsberechnung* und *Positionsbestimmung durch Engpassortung* relevant. Bei der Beacon-basierten Positionsbestimmung wird die Distanz zwischen einem mobilen Objekt und mehreren (mindestens 3) statischen Referenzpunkten berechnet. Dabei sendet entweder das mobile Objekt über einen sog. Beacon ein elektromagnetisches Signal aus, oder das Signal wird von fest an den Referenzpunkten platzierten Beacons ausgesendet. Nach dem Empfang des zuvor ausgesendeten Signals kann die Distanz des Empfängers zur Signalquelle berechnet werden, also entweder die Distanz des mobilen Senders zum Referenzpunkt oder umgekehrt. Die Berechnung der Position aus diesen (mind. 3) Distanzen kann dann über eine Triangulation erfolgen.

Problemstellung

Aufgrund der technischen Beschränkungen beim Einsatz von Outdoor-Navigationssystemen werden zur Realisierung von Anwendungsszenarien innerhalb von Gebäuden spezielle Indoor-Positionssensortechnologien benötigt. Es existiert eine große Vielfalt derartiger Technologien. Weiterhin werden Indoor-Positionssysteme (IPS) in verschiedenen Anwendungsdomänen eingesetzt. Es besteht die Aufgabe die Funktionalitäten des IPS mit den in dieser Domäne etablierten Protokollen und Systemen, bzw. mit anwendungsspezifischer Software, zu integrieren. In vielen Domänen, insbesondere auch in der Domäne der intralogistischen Anwendungen, fehlt es in bestimmten Bereichen an etablierten Standards und es besteht eine hohe technologische Entwicklungsdynamik die den häufigen Austausch von Komponenten erfordert. Allerdings ist bei der Verwendung von aktuellen Indoor-Positionssystemen der Aufwand für die Erweiterung des IPS mit neuen Komponenten, bzw. für den Austausch existierender Komponenten sehr hoch. Aktuelle Indoor-Positionssysteme weisen eine starre technologische Konfiguration auf. Dieses Problem wird durch das Bestreben von großen IPS-Herstellern verstärkt die Kompatibilität des Systems auf eigene Komponenten zu beschränken (*vendor lock-in*). Im IPos-Forschungsprojekt wird daher die Frage verfolgt nach welchem technischen Ansatz beim Entwurf von Indoor-Positionssystemen die technologische Flexibilität dieser Systeme erhöht werden kann. Konkret wird die Eignung des Framework-basierten Ansatzes untersucht. Dabei wird ein hoher Entkopplungsgrad von Komponenten angestrebt sowie die Möglichkeit zur leichten Wiederverwendung existierender Komponenten und auch häufig benötigter IPS-Funktionalitäten. Von diesen Eigenschaften des zu entwickelnden technischen Ansatzes wird eine Förderung der Agilität in der IPS-Entwicklung sowie eine Aufwandsreduktion bei der Zertifizierung wesentlicher Komponenten erwartet.

Szenarien

Die Ergebnisse des Forschungsprojektes wurden anhand von mehreren Fallstudien bewertet. Jeder Fallstudie lag dabei ein Einsatzszenario für ein Indoor-Positionssystem (IPS) zur Unterstützung einer intralogistischen Anwendung zugrunde. In der Fallstudie *Industrierobotik* wird der Schutz der Mitarbeiter im Produktionsprozess durch die Mitteilung von deren Position an einen Roboterarm erhöht. In der Fallstudie *Sensordatenfusion* wird durch das IPS aus mehreren vorhandenen Positionssensoren eines Agenten der am aktuellen Aufenthaltsort des Agenten genaueste Positionssensor ausgewählt. In der Fallstudie *Orderpicker* wird der

Kommissioniervorgang durch das IPS durch die Prüfung der korrekten Abarbeitung einer Stückliste unterstützt. In der Fallstudie *VDA5050* wird auf der Grundlage des entstandenen Frameworks ein IPS mit Unterstützung eines Ausschnitts des in der Intralogistik etablierten und standardisierten Protokolls *VDA 5050* entwickelt. In der Fallstudie *Tooz* wird auf der Grundlage des entwickelten Frameworks ein IPS mit einer Anbindung an eine Datenbrille geschaffen.

Die konzeptionelle Entwicklung der Einsatzszenarien wurde im Rahmen des Arbeitspaketes 1.3 durchgeführt.

Überblick

Dieser Abschlussbericht ist wie folgt gegliedert: Im Kapital „Architektur der Lösung“ und „Entwurf und Umsetzung der Komponenten“ werden die Projektergebnisse ausführlich vorgestellt. Dabei wird die Architektur des entstandenen Frameworks detailliert besprochen sowie der Entwurf der einzelnen Komponenten, einschließlich der entstandenen Testinfrastruktur, vorgestellt. Eine Bewertung des Projektergebnisses wird im Kapitel „Bewertung“ vorgenommen. Die Bewertung basiert auf den durchgeführten Fallstudien, die in diesem Kapitel ausführlich vorgestellt werden. Weiterhin wird das der Bewertung zugrunde gelegte Bewertungsschema beschrieben und auf das Projektergebnis angewendet. Schließlich werden in diesem Kapitel die Ergebnisse einer Anwenderumfrage vorgestellt. Kapitel „Zusammenfassung und Abschluss“ fasst den Abschlussbericht zusammen und gibt Ausblick auf mögliche weiterführende Arbeiten. Im Kapitel „Verwendung der Zuwendung“ wird die Finanzierung der geleisteten Arbeiten aus der Zuwendung dargestellt sowie die Notwendigkeit und der Nutzen dieser Arbeiten besprochen. Ein Plan zum Ergebnistransfer in die Wirtschaft wird aufgestellt.

Architektur der Lösung

Strukturprinzip

Das technische Hauptergebnis des Forschungsprojektes ist das *IPos-Framework* für die Entwicklung von Indoor-Positionssystemen. In diesem Abschnitt wird anhand von dessen Architektur ein technischer Überblick über das Framework gegeben. Die Erfüllung der im Kapitel *Problemstellung* beschriebenen Anforderungen wird anhand der Architektur diskutiert.

Eine Softwarearchitektur beschreibt ein Softwaresystem vollständig, aber grob-granular. Ein Software-Framework ist kein vollständiges Softwaresystem, es stellt aber große Teile der Architektur eines Softwaresystems bereit und definiert Möglichkeiten zu dessen Erweiterung. Ein Framework kann zu einem vollständigen Softwaresystem erweitert werden. Ergebnis dieses Projektes ist ein Software-Framework, das sogenannte IPos-FW, welches zu einem Indoor-Positionssystem erweitert werden kann. Die Architektur des IPos-FW ist daher zu unterscheiden von der Architektur eines Indoor-Positionssystems (IPS), welches mit dem IPos-FW entwickelt wurde. Ein derartiges Indoor-Positionssystem beinhaltet das IPos-FW, kann aber darüber hinaus umfangreiche funktionale Erweiterungen umfassen.

Die Softwarearchitektur des IPos-FW definiert die Struktur des Frameworks, sie identifiziert dessen Komponenten, definiert die Schnittstellen dieser Komponenten und weist den Komponenten Verantwortlichkeiten zu. Im Folgenden gehen wir auf der Grundlage von Abbildung 1 auf die Struktur des Frameworks ein. Anschließend beschreiben wir informell die

Verantwortlichkeiten der einzelnen Komponenten und gehen auf die definierten Schnittstellen ein. Die Darstellung widmet sich dabei zum einen dem IPos-FW, zum anderen aber auch den Möglichkeiten zur Erweiterung des IPos-FW zu einem Indoor-Positionssystem. Daher wird nicht nur auf die Architektur des IPos-FW eingegangen, sondern auch auf eine beispielhafte Erweiterung des IPos-FW zu einem Indoor-Positionssystem auf Grundlage einer unserer Fallstudien. Abbildung 1 beschreibt das IPos-FW (grün markierter Bereich) und dessen Erweiterung (gelb markierter Bereich) zu einem Indoor-Positionssystem gemäß unserer Fallstudie *"Orderpicker"*.

Das IPos-FW zeichnet sich durch eine gute Erweiterbarkeit und Adaptierbarkeit aus. Dieses Qualitätsmerkmal war essenziell für unsere Arbeit im Forschungsprojekt und ist durch die Architektur des IPos-FW gegeben. Um die Erweiterbarkeit und Adaptierbarkeit zu unterstützen, haben wir uns auf der Architekturebene für das Strukturprinzip der Schichtenarchitektur entschieden. Das IPos-FW ist daher bestehend aus 5 Schichten aufgebaut. Schichtenarchitekturen besitzen die folgenden Strukturmerkmale:

- Jede Schicht besitzt Verantwortlichkeiten gegenüber der jeweils über ihr liegenden Schicht.
- Im Allgemeinen bietet jede Schicht eine Schnittstelle zur darüber liegenden Schicht, sodass die obere Schicht von bestimmten technischen Details abstrahieren kann.
- Die Implementierung dieser technischen Details gehört zum Verantwortungsbereich der jeweiligen Schicht.
- Damit existiert eine Schicht, um die Funktionalität der jeweils über ihr liegenden Schicht zu ermöglichen.

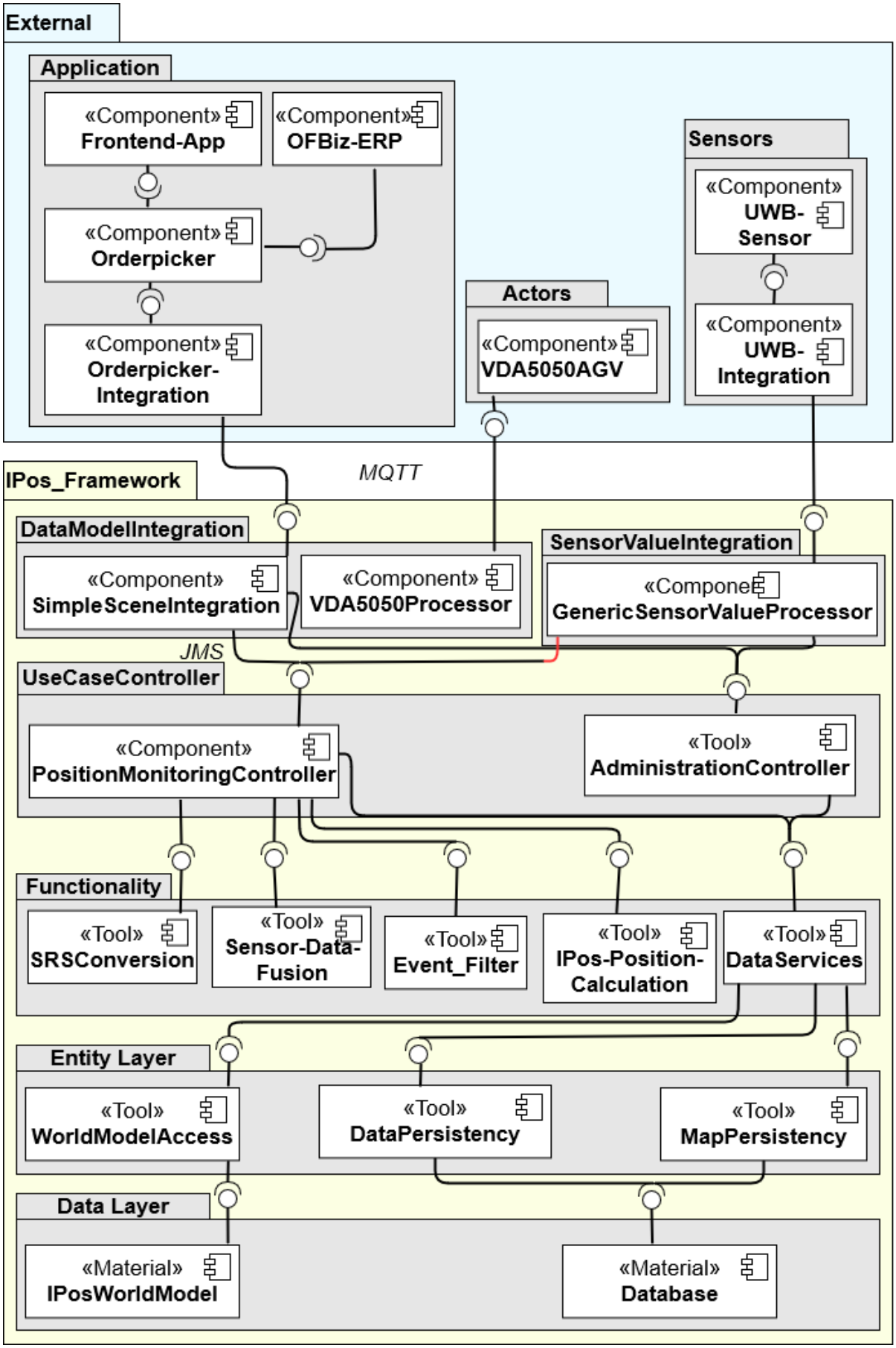

Abbildung 1 Architektur des IPos-FW

Funktionaler Überblick

Durch die Verwendung des IPos-FW als Softwarebasis erhalten Indoorpositionssysteme (IPS) eine grundlegende Anwendungsstruktur, die dem Strukturprinzip einer Schichtenarchitektur folgt. Einige Teile dieser grundlegenden Anwendungsstruktur werden bereits durch das IPos-FW mit implementierten Funktionalitäten bereitgestellt, andere Teile müssen durch die IPS-Entwickler erweitert werden. Die bereitgestellten Funktionalitäten unterstützen grundlegende Features eines Indoor-Positionssystems: Monitoring, Filterung und Persistierung. Insbesondere übernimmt ein auf Basis des IPos-FW entwickeltes Indoorpositionssystem von diesen folgenden Funktionalitäten:

- Empfang von Positionssensor-Daten
- Berechnung von Positionen aus empfangenen Positionssensor-Rohdaten
- Unterstützung verschiedener Positionssensortechnologien
- Umrechnung von Koordinatensystemen
- Selektive Weiterleitung empfangener Positionsdaten nach bestimmten Kriterien
- Persistierung empfangener Daten
- Sensordatenfusion
- Unterstützung der Austauschdatenformate Protobuf und JSON

Die Sensordatenfusionsfunktion umfasst das logische Zusammenführen von Positionssensordaten von verschiedenen Positionssensoren desselben Agenten. Die Ausnutzung der vom IPos-FW bereitgestellten Funktionalitäten erfordert die Einbettung des vom IPS-Entwickler bereitgestellten Verhaltens in die vom IPos-FW vorgegebene grundlegende Anwendungsstruktur. Der diesbezüglich verfolgte Ansatz wird weiter unten im Abschnitt "Integrationsansatz" beschrieben.

Schichten

IPos-Framework

Integrationsschicht

- Die Integrationsschicht (*Integration*) bildet die oberste Schicht des IPos-FW. Ein auf der Grundlage des IPos-FW entwickeltes Indoor-Positionssystem besteht aus dem IPos-FW und aus dem von den IPS-Entwicklern bereitgestellten Code, der über das Versenden von Anfragen an die Integrationsschicht auf das IPos-FW aufsetzt. Dem Entwurf der Integrationsschicht kommt daher eine besonders hohe Bedeutung zu, da diese die öffentliche Schnittstelle des Frameworks gegenüber seinen funktionalen Erweiterungen darstellt. Das IPos-FW wird über die Integrationsschicht zu einem Indoor-Positionssystem instanziiert.
- Die Komponenten der Integrationsschicht haben keine funktionalen Verantwortlichkeiten. Damit ist gemeint, dass diese Komponenten keine originären Funktionalitäten eines Indoor-Positionssystems implementieren. Ihre Verantwortlichkeiten liegen im Bereich der Bereitstellung geeigneter Zugriffsmöglichkeiten auf die Funktionalitäten des IPos-FW. Unter "Zugriffsmöglichkeiten" sind Operationen und erwartete Datenmodelle bzw. Datenformate für den Zugriff auf die Funktionalitäten des IPos-FW zu verstehen. Unter

einer hohen "Eignung" wird dabei ein geringer Anpassungsaufwand für existierenden Code verstanden, der auf das IPos-FW aufzusetzen ist, d. h. zu portieren ist.

- Das IPos-FW wird mit Integrationsschicht-Komponenten bereitgestellt, die den IPS-Entwicklern einen einfachen Zugriff auf das Datenmodell ermöglichen (Komponente *SimpleSceneIntegration*). Um beim Vorliegen von existierendem Code einen hohen Anpassungsaufwand zu vermeiden, ist darüber hinaus die Möglichkeit für den IPS-Entwickler vorgesehen, in der Integrationsschicht eigene Komponenten bereitzustellen. Diese Komponenten können prinzipiell in einer beliebigen Programmiersprache implementiert werden, zur Kommunikation mit den übrigen Teilen des Frameworks müssen die Komponenten allerdings in der Lage sein, als JMS-Client zu arbeiten. Während diese Bedingung für Programme erfüllt ist, die in der Java-Programmiersprache geschrieben sind, können mit Unterstützung durch geeignete Bibliotheken auch Anwendungen diese Bedingung erfüllen, die in einer anderen Programmiersprache geschrieben sind. Über die Bibliothek *Spring Python* kann diese Bedingung beispielsweise durch Anwendungen erfüllt werden, die in der Programmiersprache *Python* geschrieben sind.
- Zur besseren Unterscheidung des Verantwortungsbereiches der jeweiligen Komponenten ist die Integrationsschicht in 2 Teilschichten unterteilt. Die Komponenten der Datenmodell-Integrationsschicht (*DataModelIntegration*) ermöglichen, wie oben beschrieben, einen einfacheren Zugriff auf das Datenmodell des IPos-FW. Die Komponenten der Sensor-Integrationsschicht (*SensorValueIntegration*) übernehmen eine ähnliche Aufgabe. Sie sind verantwortlich für das Akzeptieren von Sensordaten aus einer Sensordatenquelle und für das Übersetzen dieser Sensordaten in das vom IPos-FW erwartete Sensordatenformat (Komponente *GenericSensorValueProcessor*). Ähnlich wie in der Datenmodell-Integrationsschicht können Anwender auch die Sensor-Integrationsschicht um eigene (JMS-fähige) Komponenten erweitern.

Anwendungsfallschicht

- Unmittelbar unterhalb der Integrationsschicht liegt im IPos-FW die sogenannte Anwendungsfallschicht. Ihre Komponenten stellen der Integrationsschicht eine Programmierschnittstelle auf der Abstraktionsebene konkreter Anwendungsfälle bereit. Für die weiter oben genannten Anwendungsfälle (bzw. *Features*) Monitoring, Filterung und Persistierung finden sich in der Anwendungsfallschicht spezielle Komponenten.
- Die Anwendungsfallschicht wurde zum einen eingeführt, um in der Architektur den Zusammenhang zwischen Komponenten und bereitgestellter Funktionalität übersichtlicher zu gestalten und zum anderen um die Flexibilität bei der Weiterentwicklung des IPos-FW zu verbessern. Während nämlich die relativ abstrakte Funktionalität auf der Ebene der Anwendungsfälle häufig unverändert bleiben kann, betreffen viele funktionale Überarbeitungen lediglich die darunterliegende Ebene der Implementierung dieser Anwendungsfälle (*Funktionsschicht*). Beispielsweise könnte man die Implementierung der Positionsberechnung austauschen wollen, um unterschiedliche Algorithmen miteinander zu vergleichen. Für solche Fälle ist die Aufteilung der Funktionalität in eine abstraktere (Anwendungsfallschicht) und eine konkretere Ebene (Funktionsschicht) sinnvoll.

Funktionsschicht (Functionality layer)

Genau wie im Falle der Anwendungsfallschicht liegt der Verantwortungsbereich er Komponenten der Funktionsschicht in der Bereitstellung von typischen Funktionalitäten eines IPS. Der Verfeinerungsgrad der Komponenten der Funktionsschicht ist allerdings wesentlich höher als der der Anwendungsfallschicht. Die Komponenten der Funktionsschicht stellen Funktionalitäten bereit, die von den Komponenten der Anwendungsfallschicht zur Implementierung der Anwendungsfälle benutzt werden. Die Komponenten der Anwendungsfallschicht wissen um die korrekte Anwendung der Funktionsschicht-Komponenten zur Implementierung der Anwendungsfälle. Die Komponenten der Funktionsschicht implementieren Funktionalitäten wie Berechnung von Positionen aus Positionssensor-Rohdaten oder das Treffen einer Filterentscheidung für eine empfangene Ereignisnachricht nach einem bestimmten Algorithmus.

Entitäts- und Datenschicht

Der Verantwortungsbereich der Komponenten der Entitäts- und Datenschicht liegt in der Bereitstellung von Daten gemäß dem IPos-Datenmodell. Dieses Datenmodell wird im Abschnitt „WorldModelAccess-Komponente“ genauer beschrieben. Dieser Verantwortungsbereich ist zwischen den beiden Schichten folgendermaßen aufgeteilt: Die Entitätsschicht übernimmt die konzeptionelle Bereitstellung des IPos-Datenmodells. Sie stellt der über ihr liegenden Funktionsschicht eine einheitliche Zugriffsmöglichkeit auf die gemäß des IPos-Datenmodell formulierten Daten bereit. "Einheitlich" bedeutet hier "unabhängig von der Implementierung des Datenspeichers". Die Datenschicht wiederum stellt der über ihr liegenden Entitätsschicht eine Implementierung des Datenspeichers bereit. Die Unterscheidung zwischen Entitätsschicht und Datenschicht wurde eingeführt, um die Flexibilität des IPos-FW bezüglich der technischen Realisierung der Datenbereitstellung zu erhöhen. Eine Änderung der Technologie zur Datenbereitstellung erfordert nur den Austausch der Datenschicht und Änderungen in der Entitätsschicht. Damit sind die folgenden Vorteile verbunden: (1) Es müssen keine Änderungen an den höheren Schichten (Funktionsschicht, Anwendungsfallschicht) vorgenommen werden. (2) Der einzige Zweck der Entitätschicht besteht in der Bereitstellung geeigneter Zugriffsmöglichkeiten auf das IPos-Datenmodell. Damit ist diese Schicht nicht sonderlich komplex und damit übersichtlich und gut zu warten. Die nötigen Änderungen lassen sich damit leicht identifizieren. (3) Leichte Inkompatibilitäten zwischen dem durch die Datenschicht technisch bereitgestellten Datenmodell und dem von der Entitätsschicht konzeptionell bereitgestellten Datenmodell können in der Entitätsschicht und damit an einer wohlbekannten und klar definierten Stelle überbrückt werden. Unter "Technologien zur Datenbereitstellung" verstehen wir Technologien zur Bereitstellung eines Laufzeitmodells (z. B. Laufzeitobjekte der Programmiersprache, In-Memory-Datenbank) der vorhandenen Daten und auch Persistierungstechnologien (herkömmliche Datenbankmanagementsysteme, Webservices, Einlesen von Daten von der Festplatte).

Extern

Anwendungsschicht

- Ein auf der Grundlage des IPos-FW entwickeltes Indoor-Positionssystem besteht aus dem IPos-FW und einer darauf aufsetzenden funktionalen Erweiterung. In der Schichtenarchitektur repräsentieren wir über die Anwendungsschicht diese funktionale Erweiterung. Die Anwendungsschicht enthält alle Komponenten, die in die durch das IPos-FW gegebene grundlegende Anwendungsstruktur einzubetten sind. Handelt es sich um bereits vorhandene Komponenten, die die vom Integrationlayer der IPos-FW angebotene Schnittstelle nicht

unterstützen, ist es sinnvoll, weitere Komponenten einzuführen. Ihre Aufgabe ist es, die von den vorhandenen Komponenten erwartete Schnittstelle an die von IPos-FW angebotene Schnittstelle anzupassen. Die Architektur des Beispiel-IPS in Abbildung 15 beinhaltet in der Anwendungsschicht u.a. die Komponente *Orderpicker* und die Komponente *Orderpicker-Integration*. Dieser Aufbau wird im Abschnitt „Orderpicker“ näher behandelt. Hier sei *Orderpicker* lediglich als eine beispielhafte Anwendungsschicht-Komponente angeführt, die mit dem IPos-FW zu integrieren ist. Sie wurde unabhängig von dem IPos-FW entwickelt und beinhaltet Funktionalität für die Korrektheitsprüfung des Kommissioniervorgangs. Zur Integration dieser Komponente mit dem IPos-FW wurde eine eigene Komponente *Orderpicker-Integration* geschrieben. Dieser Ansatz reduziert den Umfang der nötigen Änderungen an der existierenden Komponente (Orderpicker) zur Anpassung an das IPos-FW und wird daher empfohlen. Zudem stellt das IPos-FW für die einfachere Entwicklung derartiger Integrationskomponenten ein Entwicklungspaket zur Verfügung. Genauere Informationen können dazu im Abschnitt „Entwurf und Umsetzung der Komponenten“ gefunden werden.

Sensor-/Aktorschicht

- Ähnlich wie die Anwendungsschicht beinhaltet auch die Sensor- und die Aktorschicht externe Komponenten, die mit dem IPos-FW zu integrieren sind. Die Komponenten der Sensorschicht umfassen dabei Implementierungen unterschiedlicher Positionssensortechnologien. Zur Anpassung des Sensordatenformates an das vom IPos-FW erwartete Datenformat wird auch in unserem Beispiel zur Sensorschicht (siehe Abbildung 1) eine Integrationskomponente verwendet: Die Software des verwendeten Bluetooth-Positionssensors (Komponente *BT_Sensor*) wird über die Komponente *BT_Integration* mit dem IPos-FW verbunden.

- Die Aktorschicht kann Komponenten enthalten, die im Architekturdiagramm die Software unterschiedlicher Aktoren bspw. AGVs, repräsentieren sollen. In unserem Beispiel in Abbildung 18 beinhaltet die Aktorschicht eine Komponente, die die Software eines VDA5050-kompatiblen Aktors repräsentiert. VDA5050 ist ein Standard für die Interaktion von AGV und Leitsystem. Das IPos-FW unterstützt diesen Standard über die Komponente *VDA5050Processor* in der Integrationsschicht. Ein VDA5050-kompatibler AGV kann daher ohne eine zusätzliche Integrationskomponente mit dem IPos-FW arbeiten. Die Unterstützung des IPos-FW für den Standard VDA5050 ist ein Beispiel für die Unterstützung standardisierter Schnittstellen durch das IPos-FW. In ähnlicher Weise könnte das IPos-FW auch um die Unterstützung anderer standardisierter Schnittstellen erweitert werden. Das Lollipop/Socket-Symbol wurde in den blauen Bereich (External) gezeichnet, um zu verdeutlichen, dass das IPos-FW hier eine Schnittstelle anbietet, die konzeptionell nicht als Teil des IPos-FW entwickelt wurde, sondern von einer anderen Stelle definiert wurde. Abschnitt „VDA5050“ behandelt eine Fallstudie zur Entwicklung eines VDA5050-fähigen Indoor-Positionssystems auf der Grundlage des IPos-FWs.

Integrationsansatz

Das IPos-FW stellt das Hauptergebnis des Forschungsprojektes dar. In seiner im Rahmen des Projektes erreichten Ausbaustufe stellt es eine erweiterungsfähige, grundlegende Anwendungsstruktur von Indoorpositionssystemen (IPS) bereit. Von den Anwendern des IPos-FW (also von IPS-Entwicklern) entwickelte Erweiterungen des IPos-FW müssen in die vom IPos-FW bereitgestellte grundlegende Anwendungsstruktur eingebunden werden. Es stellt sich daher die Frage nach dem Ansatz zur Integration des IPos-FW mit seinen Erweiterungen. Die

Komponenten der Anwendungsschicht (siehe oben) werden mit den Komponenten der unter ihr liegenden Integrationsschicht (siehe oben) über eine Nachrichten-orientierte Middleware (MOM) verbunden. Aktuell wird lediglich der Nachrichtenaustausch über das Kommunikationsprotokoll MQTT unterstützt. Die Entwurfsentscheidung der Nachrichten-basierten Trennung von Framework und Erweiterungen erlaubt den IPS-Entwicklern die Erweiterung des IPos-FW unter Verwendung einer beliebigen Programmiersprache (einzige Bedingung: MQTT-Unterstützung). Diese Möglichkeit ist wichtig, um die einfache Integration verschiedener existierender Komponenten (Frontend, Logik) mithilfe des IPos-FW zu einem Indoor-Positionssystem zu unterstützen.

Durchgeführte Arbeiten und Zielerfüllung

Die Architektur des Frameworks wurde im Rahmen des Arbeitspaketes 2 "Konzept Gesamtsystemarchitektur" ausgearbeitet. Im Folgenden gehen wir näher auf die verrichteten Arbeiten ein und diskutieren für dieses Arbeitspaket die jeweils erreichte Zielerfüllung.

Die in diesem Kapitel vorgestellte Architektur ist Ergebnis umfangreicher Diskussionen und Versuche zur Untersuchung der genaueren Eignung verschiedener möglicher Implementierungstechnologien. Dazu zählt auch die Anfertigung einer prototypischen Realisierung einer frühen Version dieser Architektur. Schnittstellenspezifikationen wurden erstellt und überarbeitet. Foliensätze für mehrere Vorträge wurden ausgearbeitet. Über Diskussionen im Anschluss an die gehaltenen Vorträge fand eine Überarbeitung der Architektur statt. Im Rahmen dieses Arbeitspaketes wurden auch verschiedene Ideen für mögliche Erweiterungen des IPos-FW diskutiert.

Zur Evaluation des IPos-FW wurden in Fallstudien mehrere Indoor-Positionssysteme mit unterschiedlicher, aber relativ einfacher Funktionalität entwickelt. Diese Fallstudien werden im Abschnitt „Fallstudien“ näher vorgestellt.

Gemäß Forschungsantrag bestanden für das Arbeitspaket 2 die folgenden Ziele:

- Software-Architekturentwurf für High-Level-Schnittstellen und die Integration der Komponenten
- Modellbeschreibung in Form von DC-, Use case- und Sequenzdiagrammen; Mock-Ups für UI; Adaption an Einsatzszenarien / Demonstratoren für Logistik

Über die im Rahmen der Projektlaufzeit durchgeführten Arbeiten konnten beide Ziele vollumfänglich erreicht werden. Dazu ist im Einzelnen zu bemerken: Der entwickelte Software-Architekturentwurf sowie der verwendete Integrationsansatz wurde in diesem Kapitel beschrieben. Die High-Level Schnittstellen werden in den Abschnitten zu den Komponenten *SimpleSceneIntegration* und *GenericSensorValueProcessor* der Integrationsschicht vorgestellt und diskutiert. Modellbeschreibungen in Form von UML-Diagrammen und entwickelte Nutzeroberflächen werden in den Kapiteln zu den durchgeführten Fallstudien vorgestellt. Das IPos-FW umfasst keine Benutzeroberfläche, da diese Funktionalität stark von den konkreten Anforderungen an das IPS abhängt. Daher wurden Benutzeroberflächen nur für die Fallstudien zur praktischen Evaluation des IPos-FW entwickelt.

Simulationsumgebung

Um die Integration der Komponenten zu testen, wurde eine Simulationsumgebung implementiert. Diese Python-basierte Simulationsumgebung enthält einen Nachrichtensender,

der verschiedene Arten von Nachrichten über einen MQTT-Server an das IPos-Framework senden kann. Nachrichten, die zum Testen des Frameworks verwendet werden, sind Positionsereignisnachrichten und Rohdatennachrichten. Durch das Senden von Positionsereignisnachrichten an das IPos-FW werden die Funktionalitäten der Komponenten der Integrationsschicht getestet. Ebenso werden Rohdatennachrichten versendet, um die Integration und Funktionalität der Komponente *PositionCalculation* zu testen.

Der Code zur Implementierung des Nachrichtensenders wird auch in den Fallstudien verwendet, beispielsweise in der Fallstudie *Industrierobotik* auf dem Raspberry Pi zum Versenden von Rohdaten und in der Fallstudie *VDA5050* zum Versenden von Positionsereignisnachrichten. Die in diesem Abschnitt vorgestellten Arbeiten wurden im Rahmen des Arbeitspaketes 6.1 durchgeführt.

Verwendete Technologien

Im Folgenden geben wir einen Überblick über die getroffenen Entscheidungen bezüglich der zur Implementierung des IPos-FW verwendeten Technologien und Protokolle. Eingegangen wird auf die technische Grundlage des IPos-FW, Kommunikationsprotokolle und Indoorpositionssensoren. Die Entscheidungen werden begründet.

- Spring
    - Das IPos-FW ist eine Spring-Anwendung, d.h. es wurde als eine Instanziierung des Spring-Frameworks für die Domäne der Indoor-Positionssysteme entwickelt. Das Spring-Framework wurde aus den folgenden Gründen heraus als die Anwendungsgrundlage für das IPos-FW ausgewählt
        - Hohe Anzahl an unterstützten Bibliotheken mit einer hohen funktionalen Bandbreite (Kommunikation, Datenspeicherung, Sicherheit, Virtualisierung, etc.)
        - Aktive Nutzergemeinschaft
        - Vorhandensein eigener Erfahrungen aus vergangenen Projekten
- MQTT
    - Die Integration des IPos-FW mit den von den IPS-Entwicklern bereitgestellten Komponenten erfolgt Nachrichten-basiert über die Integrationsschicht (s.o. Abschnitt zur Integrationsschicht bzw. zum Integrationsansatz). Aktuell wird als Kommunikationsprotokoll für diese Integration nur MQTT unterstützt. Die Unterstützung weiterer Protokolle wäre wünschenswert, konnte aber aufgrund von Zeitgründen nicht realisiert werden. Ausfolgenden Gründen wurde MQTT gewählt:
        - Positive Rückmeldung der Mitglieder des projektbegleitenden Ausschusses (siehe Abschnitt „Bewertung“)
        - Geringe Leistungsanforderung, damit auch für ressourcenschwache Geräte geeignet
        - Hohe Verbreitung
- JMS
    - Die interne Kommunikation zwischen den Komponenten des IPos-FW erfolgt Nachrichten-basiert nach dem Kommunikationsprotokoll JMS. JMS wurde aufgrund seiner Ausrichtung auf die Entwicklung verteilter Java-Anwendungen gewählt. Der Einsatz von JMS ermöglicht damit die Bereitstellung von Framework-Funktionalität auf Geräte eines Indoor-Positionssystems (z.B. AGV

mit Positionssensoren, Smartphone des IPS-Anwenders). JMS wurde gewählt um diese Möglichkeit offen zu halten.

- UWB/NFC
    - Eine Übersicht der verfügbaren Indoorpositionssensortechnologien findet sich in der Arbeit (Al-Ammar [et al.] 2014). Das IPos-FW soll über seine Architektur eine Offenheit für die Unterstützung technologisch-verschiedene Indoorpositionssensortechnologien aufweisen. Für die durchgeführten Fallstudien ist daher nicht die Unterstützung spezieller Technologien wichtig, sondern die technologische Verschiedenheit der unterstützten Technologie. Aus diesem Grunde wurde eine Beacon-basierte Technologie zur Positionsbestimmung ausgewählt (UWB) sowie eine Technologie zur Engpassortung über Nahbereichskommunikation (NFC). Zum einen wurden diese beiden Technologien also aufgrund ihrer technologischen Verschiedenheit ausgewählt. Zum anderen wurden sie bei der Umfrage die wir unter den Teilnehmern des projektbegleitenden Ausschusses gemacht haben relativ häufig genannt. Eine Auswertung dieser Umfrage findet sich im Abschnitt „Anwenderumfrage“.

Entwurf und Umsetzung der Komponenten

Externe Schichten (Anwendungsschicht, Sensor- und Aktorschicht)

Die Komponenten der Anwendungsschicht stellen anwendungsspezifische Erweiterungen des IPos-FW dar. Im Rahmen des Projektes wurden mehrere Komponenten der Anwendungsschicht als Erweiterungen des IPos-FW entwickelt. Diese Komponenten wurden zur Durchführung mehrerer Fallstudien erarbeitet. Daher soll in diesem Abschnitt nicht auf den genauen Entwurf der einzelnen Komponenten der Anwendungsschicht eingegangen werden. Diese Darstellung findet sich in diesem Bericht im Kapitel „Fallstudien“ zu den einzelnen Fallstudien. Gleiches gilt für die Komponenten der Sensor- und Aktorschicht, die Komponenten zur Anbindung bestimmter Positionssensoren bzw. Aktoren an das IPos-FW beinhalten.

Im Rahmen des Forschungsprojektes sind auch technische Möglichkeiten zur Unterstützung der softwaretechnischen Entwicklung der Komponenten der Anwendungsschicht entstanden. Diese Möglichkeiten werden in diesem Abschnitt dargestellt.

Bei der Entwicklung von Indoor-Positionssystemen auf der Grundlage des IPos-FW kann der IPS-Entwickler in der Programmiersprache Java auf verschiedene Klassen und Schnittstellen zurückgreifen die die Entwicklung einer Erweiterung des IPos-FW vereinfachen. Im Einzelnen handelt es sich um die folgenden Klassen und Schnittstellen (*devkit*):

IPosExtension

Die abstrakte Klasse IPosExtension kann als Grundlage für die Entwicklung einer Erweiterung des IPos-FW verwendet werden. Von ihr abgeleitete Klassen erben von ihr einige Funktionen für die technische Realisierung der Kommunikation mit dem IPos-FW. Dabei wird für die folgende Funktionalität eine Implementierung zur Verfügung gestellt:

(1) *Schablonenmethode* mit allen Schritten für die Durchführung der Konfiguration des IPos-FW (*IPosExtension.configureIpos*)
(2) Übertragen der Konfigurationsdaten in das Austauschdatenformat (*protobuf*) und Auslösen des Kommunikationsvorgangs per MQTT (*IPosExtension.sendConfigWrapperToIpos*)
(3) Technische Ausführung der Kommunikation per MQTT über eine *geeignete Bibliothek*

Für die folgende Funktionalität ist die Entwicklung einer anwendungsspezifischen Implementierung vorgesehen. Der IPS-Entwickler muss zu dieser Implementierung bestimmte abstrakte Funktionen implementieren.

(1) *Definition* der Daten zur Konfiguration des IPos-FW (*IPosExtension.prepareConfigWrapper*)
(2) *Verarbeitung* empfangener Positionsdaten (*IPosExtension.handlePositionEvent*)
(3) *Verarbeitung* empfangener Rohdaten (*IPosExtension.handleRawdataEvent*)

eca

Das Package *eca* enthält mehrere Klassen und Schnittstellen für die Entwicklung von Algorithmen zur Verarbeitung von Positionsdaten gemäß sogenannten ECA-Regeln (ECA: *Event-Condition-Action*). Dabei wird u. a. eine *Schablonenmethode* (*ECARule.apply*) mit allen notwendigen Schritten zur Anwendung einer ECA-Regel zur Verfügung gestellt. Einige abstrakte Funktionen werden zur Implementierung durch den IPS-Entwickler definiert:

(1) *Entscheidung* bezüglich der Anwendbarkeit der Regel auf ein gegebenes Ereignis (*ECARule.isApplicable*)
(2) *Auswertung* Bedingung der Regel für das gegebene Ereignis (*Condition.evaluate*)
(3) *Ausführung* der im zweiten Schritt ermittelten Aktion, d.h. der Reaktion auf das gegebene Ereignis (*Action.execute*)

Weiterhin werden Implementierungen der oben genannten abstrakten Funktionen für eine konkrete ECA-Regel bereitgestellt. Es handelt sich dabei um eine Regel zur Überprüfung der Reihenfolge der eintreffenden Ereignisse (*ZoneSequenceRule*). Eine Erweiterung dieser Regel wird in der Fallstudie *Orderpicker* verwendet (siehe Abschnitt „Orderpicker“)

IPos2protoTransformer

Die Klasse *IPos2protoTransformer* stellt verschiedene Funktionalitäten zur Transformation der vom IPos-FW erhaltenen Positionsdaten aus dem protobuf-Austauschdatenformat in das intern verwendete Datenmodell bereit.

Abbildung 2 enthält eine Übersicht der im *devkit* enthaltenen Klassen und Schnittstellen.

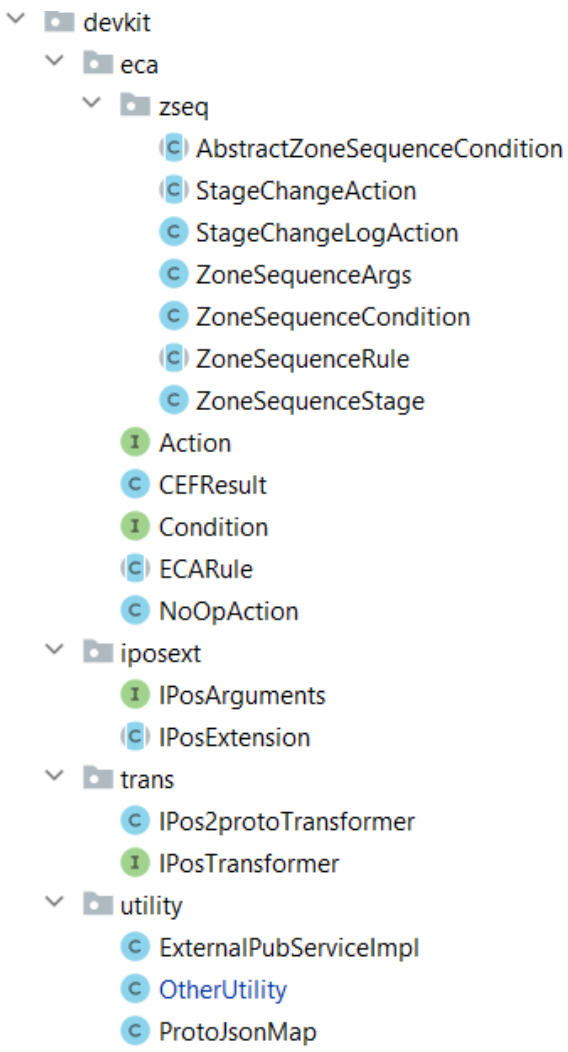

Abbildung 2 Übersicht der Klassen und Schnittstellen

Integrationsschicht

Die Integrationsschicht besteht aus den Komponenten *SimpleSceneIntegration* und *GenericSensorValueProcesso* (die in Abbildung 2 Übersicht der Klassen und Schnittstellen ebenfalls vorhandene Komponente *VDA5050Processor* wird im Abschnitt „*VDA5050“* behandelt). Erweiterungen des IPos-FW bzw. unterstützte Positionssensoren kommunizieren über diese beiden Komponenten unter Verwendung des Kommunikationsprotokolls MQTT (siehe Abschnitt „Architektur der Lösung“)

In diesem Abschnitt beschreiben wir kurz die von diesen Komponenten erwartete Schnittstelle und gehen ausführlich auf die von ihnen realisierten Verantwortlichkeiten ein. Beide Komponenten werden aufgerufen über MQTT-Nachrichten die nach den Bestimmungen des Nachrichtenaustauschformates der Datenserialisierungstechnologie *protobuf* oder des Nachrichtenaustauschformates *JSON* mit Inhalten gefüllt sind. Der Aufruf der Komponenten erfolgt damit unabhängig von der Programmiersprache in der die Komponenten implementiert sind (Java). Für IPS-Entwickler ist die von den jeweiligen Komponenten angebotene Schnittstelle relevant. Die zugehörige Schnittstellenspezifikation besteht aus einer Beschreibung von Nachrichtentypen über Ausdrücke der für das protobuf-Nachrichtenaustauschformat eingeführten Datendefinitionssprache.

SimpleSceneIntegration

Die Komponente *SimpleSceneIntegration* unterscheidet in ihrer Schnittstelle zwischen Nachrichten, die zum Zwecke der *Konfiguration* des IPos-FW empfangen werden, und solche, die zum Zwecke der *Positionsüberwachung* (*Monitoring*) abgesendet werden. In der Schnittstellenbeschreibung [SimpleSceneIntegration_proto, (IPos_GitLab n.d.)] sind die Konfigurationsnachrichten unter dem Begriff *IposConfigWrapper* zusammengefasst, die Nachrichten zur Positionsüberwachung unter dem Begriff *IposMonitoringWrapper*.

Zum einen können über Konfigurationsnachrichten dem IPos-FW Informationen über den Überwachungsgegenstand mitgeteilt werden, für den das Indoor-Positionssystem verantwortlich ist, zum anderen können konkrete Überwachungsaufträge an das Indoor-Positionssystem gegeben werden. Zur Beschreibung des Überwachungsgegenstands können POIs definiert werden (POI: point of interest; punkthafte Geoobjekte), Bezugssysteme und Raumbereiche. Ein Bezugssystem wird über einen Nullpunkt und eine Orientierung definiert. Auch können sogenannte lokalisierbare Agenten registriert werden, also beispielsweise AGVs oder Mitarbeiter. Eine genauere Beschreibung erfolgt im Abschnitt „Entitätsschicht“ anhand des vom IPos-FW verwendeten Datenmodells. Konkrete Überwachungsaufträge werden an das IPos-FW über Nachrichten vom Typ *IposMonitoringRequest* mitgeteilt. Der Überwachungsauftrag definiert das SOLL-Verhalten des IPos-FW für die Positionsüberwachung. Positionen von lokalisierbaren Agenten werden dann (d. h. gemäß einem Überwachungsauftrag) vom IPos-FW nur unter bestimmten Bedingungen (Erfüllung von *Filterkriterien*) an bestimmte Empfänger mitgeteilt. Ein Überwachungsauftrag bietet dabei die folgenden Konfigurationsmöglichkeiten:

- Mehrere Filterkriterien können spezifiziert werden (implizite logische Verknüpfung: implizites ODER)
- Gefiltert werden kann nach festen Eigenschaften der lokalisierbaren Agenten: Identifikator des Agenten, Typ des Agenten (Mensch, Roboter, Behälter, Sonstiges)
- Gefiltert werden kann nach dynamischen Eigenschaften der lokalisierbaren Agenten:
    - Aktuelle Position innerhalb eines bestimmten Raumbereiches (sog. *Zone*)
    - Distanz zwischen aktueller Position und der letzten gemeldeten Position übersteigt einen bestimmten Schwellwert (sog. *Delta*)
    - Eine bestimmte Zeitspanne ist seit der vorherigen Meldung einer Position vergangen (eine *Aktualisierungsfrequenz* kann definiert werden)
- Die vom IPos-Fw zu meldenden Informationen können inhaltlich spezifiziert werden
    - Der Gegenstand der zu meldenden Informationen kann spezifiziert werden: Position, Identifikator, Typ, Distanz. *Distanz* betrifft dabei die aktuelle Distanz des lokalisierbaren Agenten zu einem bestimmten Bezugspunkt (insbesondere zu einem bestimmten *Beacon*). Eine beliebige Kombination dieser 4 Informationsgegenstände kann ausgewählt werden.
    - Der vom IPos-FW für diesen Überwachungsauftrag zu verwendende Ansatz zur Sensordatenfusion kann gewählt werden. Als einziger Ansatz wird momentan vom IPos-FW die Weiterleitung der Position mit der höchsten Genauigkeit unterstützt. Falls ein lokalisierbarer Agent über mehrere Positionssensoren verfügt, wird dann die Position der Positionssensoren mit der momentan höchsten Genauigkeit weitergeleitet.
    - Das Bezugssystem bezüglich dessen die Positionen durch das IPos-FW zur Verfügung zu stellen sind, kann definiert werden.

- Kommunikationseigenschaften können spezifiziert werden
    - Wahl des Nachrichtenaustauschformates (protobuf oder JSON)
    - Wahl des Kommunikationsprotokolls (momentan nur MQTT, Erweiterungen möglich)
    - Der Kommunikationskanal (MQTT: sog. *topic*) über den die Positionsmeldungen vom IPos-FW zu empfangen sind, kann definiert werden

Die Abbildung 3 enthält beispielhaft die Definition des Nachrichtentyps *IposMonitoringRequest* zur Beschreibung einer Konfigurationsnachricht für die Definition eines Überwachungsauftrags. (Kaption: Nachrichtentyp zur Definition eines Überwachungsauftrages. Datendefinitionssprache der protobuf-Technologie zur Datenserialisierung)

```
message IposMonitoringRequest {
  repeated string frameIds = 1;
  float delta = 2;
  float updateFrequency = 3;
  repeated string type = 4;
  repeated string id = 5;
  string fusionStrategy = 6;
  bool exit_notification = 7;
  repeated string properties = 8;
  string monitoringTaskId = 9;
  string refSystemId = 10;
  string requestorProtocol = 11;
  string serializationType = 12;
}
```

Abbildung 3 Definition des Nachrichtentyps IposMonitoringRequest

Die Nachrichten zur Positionsüberwachung sind vom Typ *IposPositionEvent* oder *IposMsgRcvEvent*. Der Erstere enthält neben festen Informationen zum lokalisierbaren Agenten (z. B. ID und Typ) auch die aktuelle Position und Orientierung des Agenten, einen aktuellen Zeitstempel sowie eine Liste der Zonen, innerhalb derer sich der lokalisierbare Agent aktuell befindet. Die Position wird über einen Punkt in kartesischen Koordinaten (*x, y, z*) mit Genauigkeit und Bezugssystem repräsentiert, die Orientierung über eine Quaternion (*x, y, z*).

Die Komponente *SimpleSceneIntegration* realisiert die folgenden Verantwortlichkeiten:

- Empfang, Verarbeitung und Beantwortung von MQTT-Nachrichten der Anwendungsschicht nach dem unter [SimpleSceneIntegration_proto] beschriebenen Datenaustauschformat. Insbesondere Beantwortung von empfangenen Konfigurationsnachrichten (*IposConfigWrapper*) durch Nachrichten zur Positionsüberwachung (*IposMonitoringWrapper*). Empfang und Verarbeitung von Nachrichten im Datenaustauschformat JSON, sofern diese die JSON-Repräsentation von protobuf-Nachrichten entsprechen.
- Erkennen des Nachrichtentyps der empfangenen Nachricht

- Transformation der empfangenen Nachrichten aus dem Austauschdatenformat (protobuf) in das interne Objektmodell des IPos-FW unter Berücksichtigung des Nachrichtentyps der empfangenen Nachricht.
- Einbeziehung der Anwendungsfallschicht in die Erfüllung der eigenen Verantwortlichkeiten über das Versenden von JMS-Nachrichten und der Verarbeitung der erhaltenen Antwort. Dabei Berücksichtigung des zu verarbeitenden Nachrichtentyps.

GenericSensorValueProcessor

Die Komponente *GenericSensorValueProcessor* nimmt die Daten von Positionssensoren verschiedener Positionssensortechnologien entgegen. In diesem Sinne ist der Begriff *generisch* im Namen dieser Komponente zu verstehen. Die Komponente ist dafür verantwortlich, den übrigen Komponenten des IPos-FW die empfangenen Positionssensordaten im internen Objektmodell des IPos-FW zur Verfügung zu stellen. Die Komponente erwartet von den Positionssensoren über das Kommunikationsprotokoll MQTT versendete Nachrichten, deren Inhalt im protobuf-Datenaustauschformat formuliert ist. Die Kommunikation mit den Komponenten der Anwendungsfallschicht des IPos-FW erfolgt über das Kommunikationsprotokoll JMS.

Im Folgenden wird auf die von der Komponente angebotene und von den Positionssensoren zu verwendende Schnittstelle (Rohde; Zhu n.d.) eingegangen. Diese ist in der Datendefinitionssprache der Datenserialisierungstechnologie *protobuf* definiert. Die Schnittstelle definiert zwei unterschiedliche Nachrichtenformate für verschiedene Positionssensortechnologien. Die Nachrichten des ersten Formates sind im *SensorEventWrapper* zusammengefasst. Diese Nachrichten sind für die Kommunikation zwischen den Positionssensoren und dem IPos-FW gedacht. Sie enthalten nur Informationen, die ein Positionssensor selbstständig ermitteln kann, also etwa die Distanz zu einem Bezugspunkt, die Sensor-ID oder auch die Position bezüglich eines bestimmten Koordinatensystems.

Das zweite Nachrichtenformat (*IposRawdataEventWrapper*) wurde für die Kommunikation zwischen dem IPos-FW und dessen Erweiterungen eingeführt. Dieses Format unterstützt die Weiterleitung von erweiterten empfangenen Rohdaten. Insbesondere die Rohdaten bezüglich des Abstandes des Agenten von einem Bezugspunkt können für Erweiterungen interessant sein. Im Gegensatz zum ersten Format erwartet das zweite Format die Weitergabe von zusätzlichen Informationen, beispielsweise die ID des Agenten, dessen Position von den Rohdaten der Positionssensoren beschrieben wird. Die Nachrichtentypen zur Kommunikation von Positionen zwischen IPos-FW und Erweiterungen werden nicht von dieser Komponente, sondern von der Komponente *SimpleSceneIntegration* unterstützt.

Die folgenden Positionssensortechnologien werden unterstützt:

- Beacon-basiert
    - Ultrawideband (UWB)
    - Bluetooth (BT)
- Engpass-Ortung (en: proximity-based positioning)
    - RFID
    - NFC
    - Barcode

- IMU-basiert (inertiale Messeinheit; en: inertial measurement unit)

Abbildung 4 zeigt beispielhaft die Definition eines Nachrichtentyps zur Kommunikation von UWB-Rohdaten zwischen IPos-FW und Erweiterungen.

```
message IPosUWBEvent {
  string agentId = 1;
  string sensorId = 2;
  string agentType = 3;
  string sensorType = 4;
  string timeStamp = 5;
  map<string, string> distances = 6;
}
```

Abbildung 4 Nachrichtentyp (protobuf-Definition) zur Kommunikation von UWB-Rohdaten zwischen IPos-FW und Erweiterungen

Zusammenfassend lässt sich sagen, dass die Schnittstelle der Komponente *GenericSensorValueProcessor* unterscheidet zwischen

(1) Nachrichtentypen zur Kommunikation von Rohdaten zwischen Positionssensoren und IPos-FW,
(2) Nachrichtentypen zur Kommunikation von Rohdaten zwischen IPos-FW und Erweiterungen
(3) Nachrichtentypen zur Kommunikation von Positionsdaten zwischen Positionssensoren und IPos-FW

Nachrichtentypen zur Kommunikation von Rohdaten werden für die Positionssensortechnologien UWB, BT, RFID, NFC, Barcode, IMU spezifiziert.

Anwendungsfallschicht

Die Verantwortlichkeiten einer jeden Komponente der Anwendungsfallschicht bestehen in der Unterstützung eines bestimmten Anwendungsfalles. Die in der API der jeweiligen Komponente verwendeten Konzepte sind daher anwendungsfallspezifisch und eigenen sich nicht zur Wiederverwendung für andere Anwendungsfalle. In diesem Kapitel werden die beiden im Rahmen des Projektes erstellten Komponenten vorgestellt. Dabei übernimmt jede Komponente die Leitung der Datenverarbeitung des IPos-FW zur Ausführung eines bestimmten Anwendungsfalls. Alle Komponenten enden daher in ihren Bezeichnern auf *Controller*. Die Unterstützung weiterer Anwendungsfälle durch das Hinzufügen weiterer Komponente wäre bei einer Weiterentwicklung des IPos-FW möglich.

PositionMonitoringController

Die Komponente *PositionMonitoringControlle*r empfängt Nachrichten der Integrationsschicht über das Kommunikationsprotokoll JMS und greift auf die Komponenten der Funktionalitätsschicht per Java-Objektaufruf zu. Der *PositionMonitoringController* ist für den Anwendungsfall der Registrierung und Ausführung von Überwachungsaufträgen verantwortlich. Diese Verantwortlichkeiten spiegeln sich im Entwurf der Komponente wieder. Die Komponente lässt sich bezüglich ihrer inneren Struktur in 3 Bereiche unterteilen:

(1) Funktionen zur Verarbeitung von Nachrichten zur **Registrierung von Überwachungsaufträgen** (*PositionMonitoringRequests*)
(2) Funktionen zur Verarbeitung von **Rohdaten** von Positionssensoren (*RawdataEvents*)
(3) Funktionen zur Verarbeitung von **Positionen** (*PositionEvents*)

Bereich (1) erstellt für jeden empfangenen Überwachungsauftrag eine Filterkonfiguration, die von der Komponente *EventFilter* der Funktionalitätsschicht verwendet werden kann. Dabei wird für jeden Überwachungsauftrag eine eigene Instanz dieser Komponente *EventFilter* erzeugt und über die Filterkonfiguration konfiguriert und im System registriert. Erst durch die Registrierung können nachfolgende Nachrichten von Positionssensoren bezüglich aller Überwachungsaufträge ausgewertet werden. In Ergänzung zum Überwachungsauftrag wird ein sogenannter „Tracking-Auftrag“ erstellt. Aufgrund dieses Tracking-Auftrages erfolgt die Persistierung der gemäß den Kriterien des Überwachungsauftrages an den Auftraggeber weitergeleiteten Nachrichten. Diese Persistierung ermöglicht ein späteres Abrufen dieser Nachrichten und damit eine Übersicht über die Positionshistorie der Agenten.

Bereich (2) ist den Funktionen von Bereich (3) vorgeschaltet. Die empfangenen Rohdaten werden zunächst unter Verwendung von Komponenten der Funktionalitätsschicht in Positionen umgewandelt und anschließend an die Funktionen des Bereiches (3) übergeben. Für jede unterstützte Positionssensortechnologie wurde dabei eine eigene Verarbeitungslogik implementiert, die die Verarbeitung der spezifischen Rohdaten dieser Technologie ermöglicht. Für die Positionsberechnung aus den Rohdaten wird insbesondere die Komponenten *PositionCalculation* der Funktionalitätsschicht verwendet. Zur Ausführung von Überwachungsaufträgen, die eine Weiterleitung von Rohdaten vorsehen, werden im Bereich (2) auch Rohdaten an die Auftraggeber weitergeleitet und dazu wieder an die Integrationsschicht übergeben.

Bereich (3) empfängt Positionen zur Weiterverarbeitung und kann dabei nicht unterscheiden, ob die Positionen ursprünglich von einem Positionssensor gesendet wurden oder zuvor innerhalb der Funktionen des Bereiches (2) des *PositionMonitoringControllers* berechnet wurden. Eine für deren Verarbeitung wichtige Eigenschaft von Positionen ist das Bezugssystem bezüglich, dessen sie formuliert sind. Um in diesem Punkt eine einheitliche Behandlung zu ermöglichen, wurde im Entwurf festgelegt, dass der Bereich (3) des *PositionMonitoringControllers* ausschließlich mit Positionen arbeitet, die bezüglich des Standard-Koordinatensystems des IPos-FW (*ROOT*, siehe Komponente *SRSConversion* der Funktionalitätsschicht) definiert sind. Im ersten Schritt erfolgt daher eine Transformation der erhaltenen Position in dieses Koordinatensystem. Die Transformation erfolgt unter Verwendung der Komponente *SRSConversion* der Funktionalitätsschicht. Im zweiten Schritt erfolgt eine Bestimmung der für das Treffen einer Entscheidung zur Sensordatenfusion relevanten Eigenschaften der Position. Dabei wird insbesondere festgestellt, ob zum jeweiligen Zeitpunkt für diesen Agenten eine aktuelle Position mit einer höheren Genauigkeit von einem anderen Positionssensor verfügbar ist. Diese Entscheidung wird unter Verwendung der Komponente *SensorDataFusion* der Funktionalitätsschicht getroffen. Schließlich wird die Relevanz der empfangenen Position für jeden der zuvor registrierten Überwachungsaufträge berechnet. Diese Berechnung erfolgt unter Verwendung der dem jeweiligen Überwachungsauftrag zugeordneten Instanz der *EventFilter*-Komponente der Funktionalitätsschicht. Im Ergebnis der Filterung wird diese entweder im Hinblick auf den jeweiligen Überwachungsauftrag ignoriert oder sie wird an die Integrationsschicht zurückgegeben, um von dieser an den jeweiligen Auftraggeber des Überwachungsauftrages weitergeleitet zu werden. Das Ignorieren erfolgt

dabei aus zwei möglichen Gründen, entweder, weil sie die für diesen Auftrag definierten Filterbedingungen selbst nicht erfüllt oder weil bei Vorliegen des Kriteriums *Sensordatenfusion* eine andere Nachricht die Kriterien besser erfüllt. Des Weiteren erfolgt bei der positiven Auswahl der Positionsnachricht durch eine EventFilter-Instanz die Persistierung der Nachricht für den Überwachungsauftrag der der EventFilter-Instanz zugeordnet ist. Der Anwendungsfall zur Positionsüberwachung umfasst ebenfalls die Funktionalität den Auftraggeber nicht nur über die aktuelle Position eines lokalisierbaren Agenten, sondern auch über dessen Eintreten in eine bestimmte Zone oder über dessen Austreten aus dieser Zone. Daher erfolgt bei positiver Auswahl der Positionsnachricht durch einen EventFilter zusätzlich eine Aktualisierung der Zonenzugehörigkeit des dem Positionssensor zugeordneten lokalisierbaren Agenten gemäß der für den jeweiligen Überwachungsauftrag relevanten Zonen.

Das UML-Sequenzdiagramm in Abbildung 5 beschreibt den Ablauf der Auswertung einer eingehenden Nachricht mit Rohdaten eines Positionssensors unter Berücksichtigung der registrierten Überwachungsaufträge. Zunächst wird aus den erhaltenen Rohdaten die Position berechnet. Dies geschieht unter Rückgriff auf eine Komponente aus der Funktionalitätsschicht (*PositionCalculation*, siehe folgender Abschnitt). Anschließend erfolgt eine Auswertung der berechneten Position bezüglich der Kriterien zur Sensordatenfusion. Eine Nachricht, die gemäß diesen Kriterien nicht weiterverarbeitet werden soll (also nicht zum Bestand der fusionierten Daten gehört) wird ignoriert (siehe *else*-Alternative). Konkret wird überprüft, ob die Position von einem Positionssensor gesendet wurde, der einem lokalisierbaren Agenten zugeordnet ist für den zu diesem Zeitpunkt noch zusätzlich aktuelle Positionsdaten von einem Positionssensor mit einer höheren Genauigkeit vorliegen. Für eine demnach relevante Position wird anschließend jeder der registrierten Überwachungsaufträge (*loop*-Bereich) ausgewertet. Jedem Überwachungsauftrag ist dabei eine EventFilter-Instanz zugeordnet. Akzeptiert die EventFilter-Instanz die Positionsnachricht, wird diese an den Auftraggeber des Überwachungsauftrages weitergeleitet, der der jeweiligen EventFilter-Instanz zugeordnet ist. Falls die EventFilter-Instanz die Positionsnachricht nicht akzeptiert, wird sie für den jeweiligen Überwachungsauftrag ignoriert (*else*-Bereich).

AdministrationController

Die Komponente *AdministrationController* implementiert den Anwendungsfall der Administration. Dieser Anwendungsfall umfasst verschiedene Aufgaben, die der Bereitstellung der übrigen Funktionalitäten des IPos-FW dienen. Sie wird von den Komponenten der Integrationsschicht und der Komponente *PositionMonitoringController* der Anwendungsfallschicht benutzt. Der Komponente *SimpleSceneIntegration* bietet sie die Möglichkeit zur Initialisierung des Systems an. Zur Implementierung dieser Funktionalität wird die Komponente *DataServices* der Funktionalitätsschicht benutzt. Die zweite Aufgabe dient der Durchsetzung eines einheitlichen Bezugssystems bei der Positionsverarbeitung innerhalb des IPos-FW. Sie bietet dem PositionMonitoringController und den Komponenten der Integrationsschicht die Möglichkeit an empfangene Positionen in das ROOT-Koordinatensystem zu transformieren. Zur Implementierung dieser Funktionalität wird die Komponente *SRSConversion* der Funktionalitätsschicht verwendet.

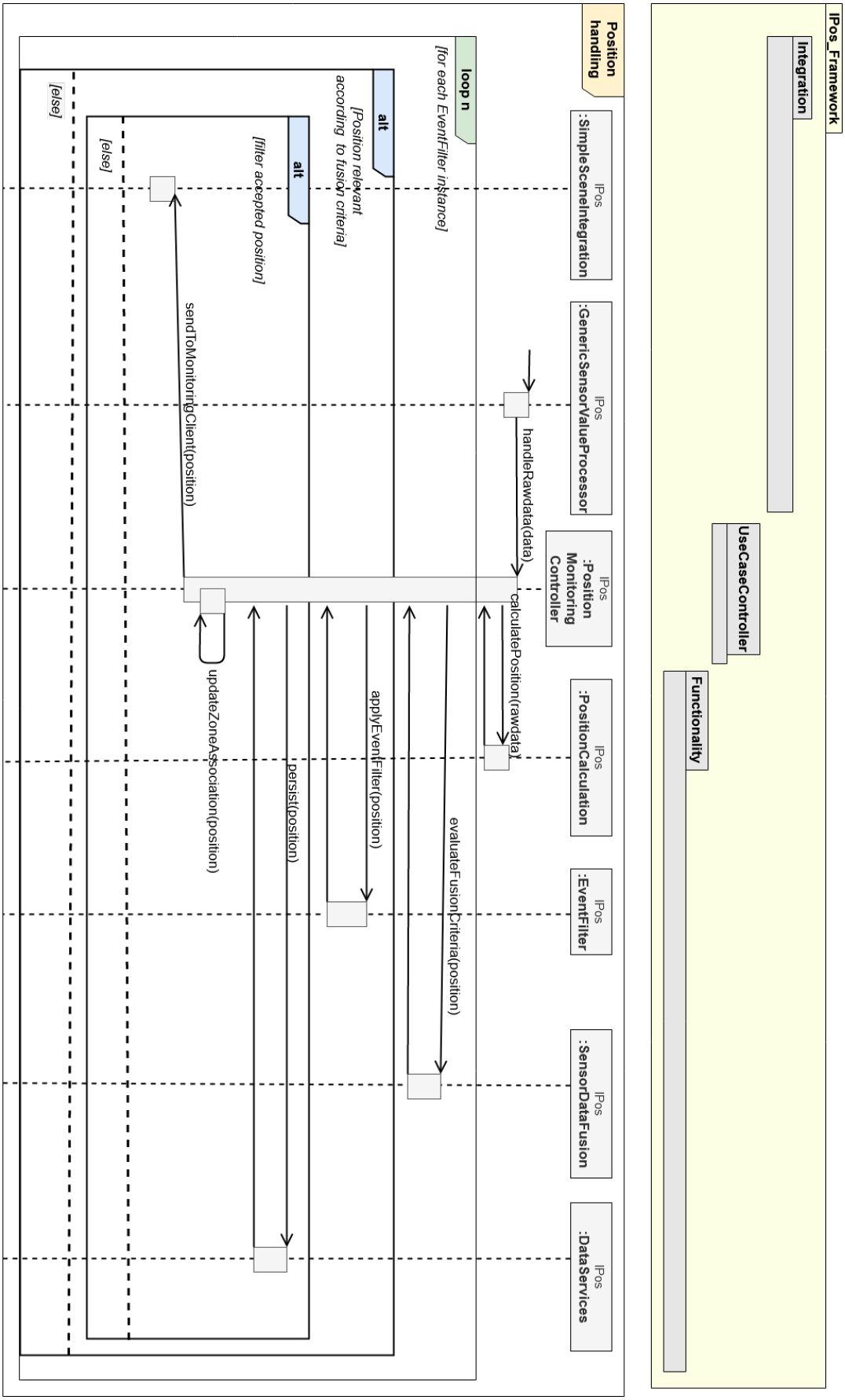

Abbildung 5 UML-Sequenzdiagramm zur Auswertung eingehender Nachrichten unter Berücksichtigung der registrierten Überwachungsjobs

Funktionalitätsschicht

SRSConversion

In verschiedenen Indoor-Agenten und -Systemen werden unterschiedliche (Sub-)Koordinatensysteme verwendet. Die Komponente *SRSConversion* implementiert die Koordinatenkonvertierung zwischen verschiedenen Koordinatensystemen. Im IPOS-Framework werden die Beziehungen zwischen verschiedenen Koordinatensystemen als Quaternion auf dem Weltmodell gespeichert. SRS Conversion hat zwei Hauptfunktionen: Die erste Funktion ist die Berechnung der Koordinate im Zielkoordinatensystem aus einer Koordinate im Originalsystem, die zweite die Berechnung der neuen Quaternion, wenn das Bezugssystem eines Koordinatensystems geändert wird.

EventFilter

Die Komponente *EventFilter* ist ein zentraler Bestandteil des Frameworks. Der Eventfilter entscheidet anhand der Filterkonfigurationen, ob und wohin ein Event weitergeleitet werden soll. Der EventFilter ist ein boolescher Prädiktor, die Eingabe davon ist ein positionEvent-Objekt, die Ausgabe ist ein boolescher Wert (0 für Pass, 1 für Block). Die Bedingungen des Filters können über die Konfigurationsschnittstelle konfiguriert werden: einzelner Parameter oder vollständige Konfigurationsdatei. Zur Unterstützung der Kernfunktionen des IPOS-Frameworks sind insgesamt 8 Arten von Filterbedingungen definiert:

Tabelle 1 Filterbedingungen des Eventfilters

Filterbedingung	Type der Bedingung
Zeit	Übereinstimmung oder Intervall
Kategorie	Übereinstimmung
Position	Entfernungs- oder Gebietsabhängig
Agent ID	Übereinstimmung
Sensor ID	Übereinstimmung
Genauigkeit	Grenzwert
Zeit des minimalen Aktualisierungsintervalls	Grenzwert
Position Delta	Grenzwert

Die letzten beiden Filterbedingungen sind Stream-basierte Bedingungen. Sie definieren das minimale Zeit- und Raumintervall zwischen zwei aufeinanderfolgenden Events. Ein Filter kann mehrere Bedingungstypen haben, und die Beziehung zwischen den Bedingungstypen ist verkettet, was bedeutet, dass das Event alle Bedingungen erfüllen muss, um den Filter passieren zu können.

```
{
  "time_condition": [
    ["2021-07-05 14:09:03.591", "2021-07-06 14:09:03.591"],
    ["2018-09-05 14:00:00.000", "2018-09-05 15:00:00.000"]
  ],
  "category_condition": [
    "human",
    "AGV"
  ],
  "id_condition": [
    "AGV001"
  ],
  "sensor_id_condition": [
    "UWB*",
    "SLAM*"
  ],
  "accuracy_condition": [
    0.5
  ],
  "position_condition": [
    [2, 3, 1, 1],
    [1, 2, 1, 1]
  ],
  "time_min_interval": [
    1000
  ],
  "position_delta": [
    1
  ],
  "position_condition_cell": [
    [[1, 2, 3], [4, 5, 6], [7, 8, 9, 10]],
    [[11, 12, 13], [14, 15, 16], [17, 18, 19, 20]]
  ]
}
```

Abbildung 6 Beispiel einer Konfigurationsdatei eines EventFilters

Abbildung 6 zeigt ein Beispiel für die Konfiguration eines Filters. In einigen der Bedingungskategorien, wie z. B. Agentenkategorie (category_condition) und Sensortypen (sensor_id_condtion), ist eine Liste von Bedingungen definiert. Die Bedingungen in derselben Liste sind parallel. Mit diesem Setup benötigen Aufgaben wie „Positionsaktualisierungen von AGVs und Menschen von gestern in einem bestimmten Bereich abrufen“ nur einen EventFilter.

PositionCalculation

Wenn das IPOS-Framework Rohdaten von Sensoren empfängt, müssen sie in "Positionsevents" übertragen werden. Die Komponente *PositionCalculation* implementiert Funktionalitäten der Positionsberechnung. Das Format der Rohdaten hängt von den Positionstechnologien ab, es werden drei Arten von Technologien unterstützt: Beacon-basierte, Engpass-basierte und IMU-basierte Technologien. Basierend auf dem vorgeschlagenen Datenmodell kann die Erweiterung auf andere Arten von Technologien unterstützt werden.

SensorDataFusion

Die Komponente *SensorDataFusion* implementiert Funktionalitäten der Sensordatenfusion. Sie ermöglicht die Verknüpfung der Positionsdaten unterschiedlicher Positionssensoren zur Steigerung der Qualität der bereitgestellten Positionsdaten. Konkret zielt diese Komponente auf die Verbesserung der Genauigkeit der für einen bestimmten lokalisierbaren Agenten bereitgestellten Positionsdaten. Sie erkennt das Vorliegen von Positionsdaten mehrerer unterschiedlicher Positionssensoren für ein und denselben Agenten. Zur Verbesserung der

Genauigkeit der für diesen Agenten bereitgestellten Positionsdaten ist diese Komponente in der Lage, ungeeignete Positionsdaten zu verwerfen. Verworfen werden dabei alle Positionsdaten außer den Positionsdaten der Positionssensoren mit der höchsten Messgenauigkeit. Das Alter der jeweiligen Positionsdaten wird bei dieser Entscheidung mitberücksichtigt.

DataServices

Die Komponente *DataServices* stellt für die Komponenten der höheren Schichten des IPos-FW einen Zugriffspunkt auf die dem IPos-FW aktuell vorliegenden Daten zur Verfügung. Das zu deren Beschreibung verwendende Datenmodell wird im Abschnitt zur Komponente *WorldModelAccess* genauer beschrieben. Bereitgestellt werden Funktionen zum Hinzufügen von einzelnen Datenpunkten und zum selektiven Abruf von Datenpunkten.

Entitätsschicht

WorldModelAccess

Die Verantwortlichkeit der Komponente *WorldModelAccess* besteht in der Laufzeitdatenhaltung des IPos-FW. Sie stellt die Laufzeitdaten nach dem in Abbildung 7 beschriebenen Datenmodell zur Verfügung. Auf diese Komponente wird ausschließlich über die Komponente *DataServices* der Funktionalitätsschicht zugegriffen. Im Folgenden wird auf das von der Komponente *WorldModelAccess* bereitgestellte Datenmodell genauer eingegangen.

Das Datenmodell ist in Form von UML-Klassendiagrammen dargestellt. Das Modell lässt sich konzeptionell in folgende Bereiche unterteilen:

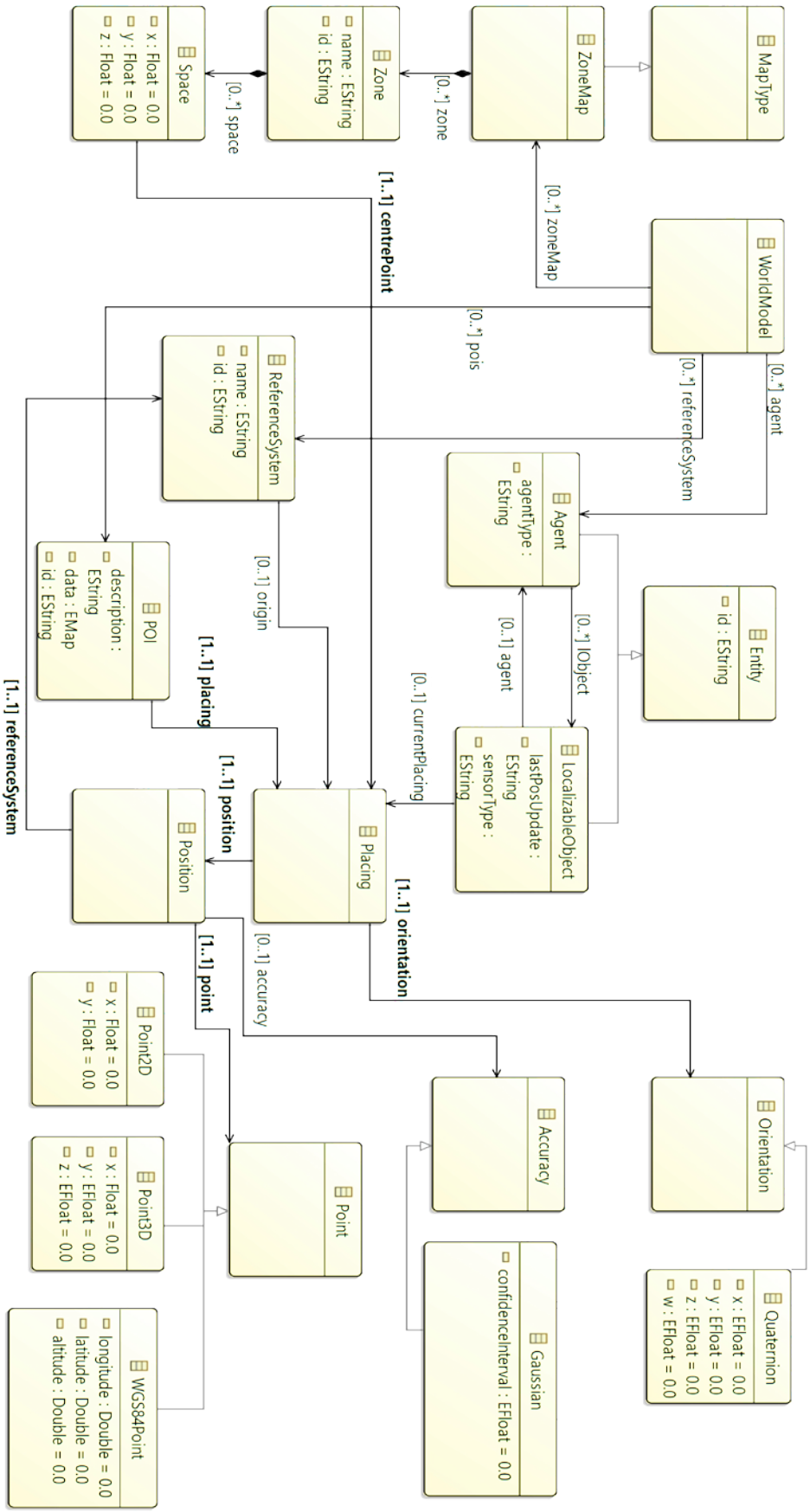

Abbildung 7 Agentenbeschreibung nach dem IPos-Datenmodell

Agentenbeschreibung

Der Bereich *Agentenbeschreibung* umfasst alle Klassen, über den positionsbezogenen Informationen zu den Agenten abgebildet werden können. Agenten werden als mobile Entitäten betrachtet, deren Position vom Indoor-Positionssystem zu erfassen ist.

Da ein Indoor-Positionssystem in erster Linie für die Erfassung von positionsbezogenen Informationen verwendet wird, wurde das Datenmodell konzeptionell darauf beschränkt. Es gibt sehr viele weitere Informationen, über die Agenten näher beschrieben werden können. Diese können über teils standardisierte, domänenspezifische Datenmodelle abgebildet werden. In der Fallstudie *Orderpicker* wird gezeigt, wie durch eine Erweiterung des IPos-FW das IPos-Datenmodell mit domänenspezifischen Datenmodellen verknüpft werden kann.

Das IPos-Datenmodell umfasst neben der reinen Positionsbeschreibung auch eine Beschreibung der Orientierung des Agenten sowie Informationen zu verwendeten Bezugssystemen und Karteninformationen.

- *Agent:* Agenten werden als mobile Entitäten betrachtet, deren Position vom Indoor-Positionssystem zu erfassen ist. Die Klasse *Agent* repräsentiert einen Agenten mit einer Identität, einem Typ sowie mehreren Positionssensoren. Jede Instanz der Klasse *LocalizableObject* repräsentiert einen Positionssensor einer bestimmten Technologie, der dem Agenten zugeordnet ist. Agententypen sind z. B. BOX, BIN, ZONE, ROBOT, HUMAN, OTHER. Das IPos-Datenmodell sieht keine Beschränkungen des Wertebereiches der Agententypen vor. Die möglichen Sensortypen sind auf die vom IPos-FW unterstützten Positionssensortechnologien beschränkt: BLUETOOTH, UWB, NFC, RFID, BARCODE, OTHERBEACON, OTHERPROXIMITY.
- *Positionsbeschreibung:* Jedem Positionssensor eines Agenten, d. h. jedem lokalisierbaren Objekt (Klasse *LocalizableObject*) ist eine Position und eine Orientierung zugeordnet. Beide Informationen werden unter dem Begriff "Platzierung" (Klasse: *Placing*) zusammengefasst. Eine Instanz der Klasse *Placing* ist also konzeptionell ein Aggregat aus einer Position und einer Orientierung.
- *Positionen:* Positionen bestehen aus einem Punkt (Klasse: *Point*), einem Bezugssystem und einer Genauigkeitsinformation. Da je nach verwendetem Bezugssystem zur Beschreibung von Punkten unterschiedliche Informationen benötigt werden, wurde im IPos-Datenmodell die Möglichkeit vorgesehen, die Klasse *Point* zu spezialisieren und Punkte über unterschiedliche Klassen zu beschreiben. Das IPos-Modell unterstützt auf dem aktuellen Stand zweidimensionale und dreidimensionale Punkte in kartesischen Koordinaten sowie Punkte im Koordinatensystem "WGS84".
- *Orientierungen:* Das IPos-Datenmodell sieht grundsätzlich die Möglichkeit vor Orientierungen (Klasse: *Orientation*) auf unterschiedliche Arten näher zu beschreiben. Obwohl Erweiterungen hier möglich wären, bietet das IPos-Datenmodell auf dem aktuellen Stand lediglich die Beschreibung von Orientierungen über Quaternionen an (Klasse: *Quaternion*). Eine Erweiterung erschien bisher nicht notwendig, da die Beschreibung von Orientierungen über Quaternionen etabliert ist und von existierenden Algorithmen zur Koordinatentransformation zwischen Bezugssystemen unterschiedlicher Orientierung unterstützt wird.
- *Bezugssysteme:* Das IPos-Datenmodell erlaubt die Definition verschiedener Bezugssysteme. Überwachungsaufträge können die Bereitstellung von Positionen in einem bestimmten Bezugssystem vorsehen. Bekannte Positionen können von

Positionssensoren oder anderen Anwendungen in einem bestimmten Bezugssystem an das IPos-FW gesendet werden. Verwendet werden können dabei nur Bezugssysteme, die zuvor beim IPos-FW registriert wurden. Bezugssysteme haben eine Id und ihnen kann ein Ursprung zugeordnet sein. Das IPos-FW besitzt ein vordefiniertes Bezugssystem: ROOT. Diesem Bezugssystem ist kein Ursprung zugeordnet. Der Ursprung des Bezugssystems besteht aus einer Orientierung und einer Position. Die Position ist ebenfalls bezüglich eines bestimmten Bezugssystems definiert. Es ergibt sich eine Verkettung verschiedener Bezugssysteme. Eine Baumstruktur kann gebildet werden. Die Wurzel des Baumes ist das ROOT-Bezugssystem. Für die initiale Konfiguration des IPos-FW sollte an einer festen Stelle im vom IPS abgedeckten Innenraumes der Ursprung des ROOT-Systems angenommen werden. Die Ursprungskoordinaten anderer Koordinatensysteme können dann relativ zu dieser festen Stelle angegeben werden.

- *Karteninformationen*: Das IPos-Datenmodell unterstützt das Anlegen von Informationen über die Umgebung, in der sich die mobilen Agenten bewegen können. Die Klasse *MapType* dient dabei allen Klassen zur Beschreibung einer Umgebungskarte als Oberklasse. Im Rahmen des Forschungsprojektes konnte lediglich ein Kartentyp angelegt werden: Die Zonenkarte (Klasse: *ZoneMap*). Eine Karte dieses Typs besteht aus einer Mehrzahl von Zonen, wobei jede Zone aus mehreren Raumbereichen (Klasse: *Space*) besteht. Ein Space ist ein quaderförmiger Raumbereich mit einem bestimmten Mittelpunkt (Assoziation: *centrePoint*). Die Datenstruktur *Zone* hat eine große Bedeutung für Überwachungsaufträge, da das Weiterleiten von Agentenpositionen an den Aufenthalt des Agenten innerhalb einer bestimmten Zone gekoppelt werden kann. Außerdem können Umgebungsinformationen über sogenannte Points-of-Interest (POI) abgelegt werden. Ein POI ist ein identifizierter Punkt mit einer Orientierung und weiterführenden Informationen.
- *WorldModel*: Die Klasse *WorldModel* besitzt als ein wichtiger Knotenpunkt im Datenmodell Zugriff auf alle positionsbezogenen Informationen zu den Agenten. Sie wird von der Komponente *DataService* für den Abruf der Informationen verwendet und stellt dafür geeignete Abrufmethoden bereit.

Ereignisse

Als Ereignis wird ein *Zustandswechsel von Interesse* verstanden. Für ein Indoor-Positionssystem sind nur Zustandswechsel mit Positionsbezug von Interesse. Daher unterstützt das IPos-FW u. a. die Behandlung von Positionsereignissen und Rohdatenereignissen von Positionssensoren über die Definition geeigneter Ereignisarten. Im Folgenden werden die unterschiedlichen vom IPos-FW unterstützten Ereignisarten näher betrachtet:

- *RawdataEvent:* Die Weitergabe von Positionssensor-Rohdaten an das IPos-FW kann über sogenannte *RawdataEvents* erfolgen. Ein Rohdatenereignis wird versendet, wenn ein Indoor-Positionssensor neue Rohdaten ermittelt hat und nicht selbst daraus eine Position bestimmen kann, sondern die Positionsberechnung bzw. Weiterverarbeitung dem IPS zu überlassen hat. Aufgrund großer Unterschiede im Messprinzip unterscheiden sich auch die von den unterschiedlichen Positionssensor-Technologien bereitgestellten Rohdaten deutlich. Das IPos-FW unterstützt daher technologiespezifische Formate für die Rohdatenereignisse der unterschiedlichen

Positionssensor-Technologien. Allen Formaten gemeinsam ist lediglich das Vorhandensein eines Zeitstempels (*timestamp*).

- *Beacon:* Es werden Sensorrohdaten-Formate für unterschiedliche Beacon-basierte Positionssensortechnologien unterstützt. Allen technologiespezifischen Formaten sind die folgenden Eigenschaften gemeinsam: Als Kerninformation wird eine Abbildung von Beacon-ID auf Distanz mitgeteilt. Die *Distanz* ist dabei die gemessene Entfernung des Beacons mit der *Beacon-ID* vom Positionssensor. Außerdem wird die Id der Positionssensoren mitgeteilt und der Name der Positionssensortechnologie (*Typ*). Als spezielle Beacon-basierte Positionssensortechnologien werden *Bluetooth* und *UWB* unterstützt. Das UWB-spezifische Format unterscheidet sich nicht vom allgemeinen Format für Beacon-basierte Positionssensortechnologien. Das Bluetooth-spezifische Format erlaubt darüberhinausgehend die Kommunikation von einer Abbildung von Beacon-IDs auf den Wert der empfangenen Signalstärke (*rss*). Zur vorausschauenden Unterstützung von Rohdatenformaten zukünftiger, heute noch unbekannte Positionssensortechnologien wurde auch die Unterstützung des Datenformates *OtherBeacon* vorgesehen. Dieses Format enthält alle Informationen des allgemeinen Formates für Beacon-basierte Positionssensortechnologien sowie darüberhinausgehend eine Abbildung von Attributnamen auf Attributwerte für die Kommunikation von beliebigen Schlüssel-Wert Paaren.
- *IMU:* Das IPos-FW unterstützt ein Format für Sensorrohdaten-Ereignisse von inertialen Messeinheiten. Demnach kombinieren IMU-Rohdaten Ereignisse Informationen zur Linearbeschleunigung als auch zur aktuellen Drehrate des Agenten.
- *Proximity:* Es werden Sensorrohdaten-Formate für unterschiedliche Positionssensortechnologien der Engpassortung unterstützt. Allen technologiespezifischen Formaten sind die folgenden Eigenschaften gemeinsam: Als Kerninformation wird die Id des vom Scanner erkannten Tags sowie die Id des Scanners mitgeteilt. Außerdem wird der Name der Positionssensortechnologie (*Typ*) mitgeteilt. Die folgenden Positionssensortechnologien zur Engpassortung werden unterstützt: *RFID*, *NFC*, *Barcode*. Das Barcode-spezifische Format unterscheidet sich nicht vom allgemeinen Format für Positionssensortechnologien der Engpassortung. Das RFID- bzw. NFC-spezifische Format unterstützt über das String-Attribut *location* bzw. die Map-Struktur *tagData* die Möglichkeiten dieser Positionssensortechnologien zur Ablage sonstiger Daten. Zur vorausschauenden Unterstützung von Rohdatenformaten zukünftiger, heute noch unbekannte Positionssensortechnologien wurde auch die Unterstützung des Datenformates *OtherProx* vorgesehen. Dieses Format enthält alle Informationen des allgemeinen Formates für Positionssensortechnologien der Engpass-Ortung sowie darüberhinausgehend eine Abbildung von Attributnamen auf Attributwerte für die Kommunikation von beliebigen Schlüssel-Wert Paaren.
- *PositionEvent:* Das IPos-Datenmodell sieht auch ein Format für Rohdatenereignisse für solche Positionssensoren vor die in der Lage sind Positionen zu berechnen. Derartige *PositionEvents* kommunizieren neben der aktuellen Position und der aktuellen Orientierung auch die ID der Sensoren und einen aktuellen Zeitstempel. Diese Informationen werden bei der Verarbeitung durch das IPos-FW um weitere Informationen ergänzt. Dazu zählt die aktuelle Zonenzugehörigkeit des Agenten (die aus der Position und die bekannten Lagen der Zonen berechnet werden kann).
- *MessageReceivedEvent*: Zur Unterstützung der Weiterleitung bzw. anwendungsspezifischen Verarbeitung von Nachrichten sonstiger Protokolle die von den Agenten verwendet werden (z. B. *VDA5050* als Protokoll für die Kommunikation von

AGVs und AGV-Steuerungen) wurde das Rohdatenformat *MessageReceivedEvent* eingeführt. Dieses unterstützt zum einen die Weitergabe empfangener Nachrichten in serialisierter Form (*serializedMsg*) und zum anderen die Weitergabe von Attributen, die aus der empfangenen Nachricht extrahiert wurden. Die aktuelle Position und Orientierung kann (falls bekannt) ebenfalls über ein eigenes Feld kommuniziert werden.

Eine grafische Darstellung des Ereignis-Ausschnittes des IPos-Datenmodells findet sich in der Abbildung 9. Aus der Abbildung wird deutlich, dass die beschriebenen Datentypen im IPos-Datenmodell in zwei Varianten existieren. Die Variante des Packages *iPos_Datamodel* wird vom IPos-FW zur Abbildung der von den Positionssensoren empfangenen Daten verwendet. Dieses Format enthält daher nur Informationen, die von dem Positionssensor geliefert werden können. Die Varianten im Package *IPosDevKit* enthalten alle Informationen, die vom IPos-FW an die Auftraggeber von Monitoring-Aufträgen bereitgestellt werden. Sie werden also für die Abbildung der bereits durch das IPos-FW erweiterten Daten verwendet. Diese Variante geht inhaltlich über die Datentypen der ersten Variante hinaus und sieht auch Informationen zur Agenten-ID und Agenten-Typ sowie zur aktuellen Zonenzuordnung des Agenten vor.

Systemsteuerung und Konfiguration:

Das IPos-Datenmodell beinhaltet auch Datenstrukturen zur Systemsteuerung und Konfiguration. Diese Datenstrukturen dienen der Abbildung und Verarbeitung von empfangenen Anfragen. Die Filterkriterien eines Überwachungsauftrages (*MonitoringRequest*) wurden bereits im Abschnitt „Integrationsschicht“ näher beschrieben. Ein MonitoringRequest wird dabei auf die Datenstruktur *MonitoringRequest* (Abbildung 10) abgebildet. Bei der Liste *frameid* handelt es sich um eine Liste von Identifikatoren der für diesen Überwachungsauftrag zu überwachenden Raumbereiche (Zonen). Das Feld *Delta* erfasst einen für die Mitteilung von Positionsänderungen relevanten Schwellwert. Das Feld *updateFrequency* erfasst eine für die Mitteilung von Positionsänderungen relevante Aktualisierungsfrequenz. Die Felder *type* und *id* erfassen Bedingungen bezüglich der Typen und Identifikatoren der zu überwachenden Agenten. Das Feld *fusionStrategy* erfasst den für diesen Überwachungsauftrag zu verwenden Ansatz zur Sensordatenfusion. Über das Feld *exitNotification* wird konfiguriert, ob der Überwachungsauftraggeber über das Eintreten des Agenten in eine Zone bzw. über das Heraustreten des Agenten aus einer Zone informiert werden soll. Über das Feld *properties* wird der Gegenstand der zu meldenden Information mitgeteilt (Position, Identifikator, Typ, Distanz). Die Felder *monitoringTaskId*, *requesterProtocol*, *serializationType*, *refSystemId* setzen Eigenschaften des Nachrichtenaustausches fest. Der Name des zu verwendenden Kommunikationskanales (MQTT: Topic) ist dabei die monitoringTaskId. Das requesterProtocol setzt das vom Überwachungsauftraggeber erwartete Kommunikationsprotokoll fest (unterstützt wird auf dem Berichtsstand lediglich MQTT). Als Serialisierungsformat wird protobuf und JSON unterstützt. Das Feld *refSystemId* setzt das Bezugssystem fest, bezüglich dessen der Überwachungsauftraggeber die gesendeten Positionen erwartet. Abbildung 10 zeigt ebenfalls die Datenstruktur EventFilterCondition. Die EventFilter des IPos-FW sind in der Lage, Filterbedingungen aus dieser Datenstruktur herauszulesen. Einige der vom Überwachungsauftraggeber über einen *MonitoringRequest* mitgeteilten Informationen werden daher in eine Instanz dieser Datenstruktur übernommen.

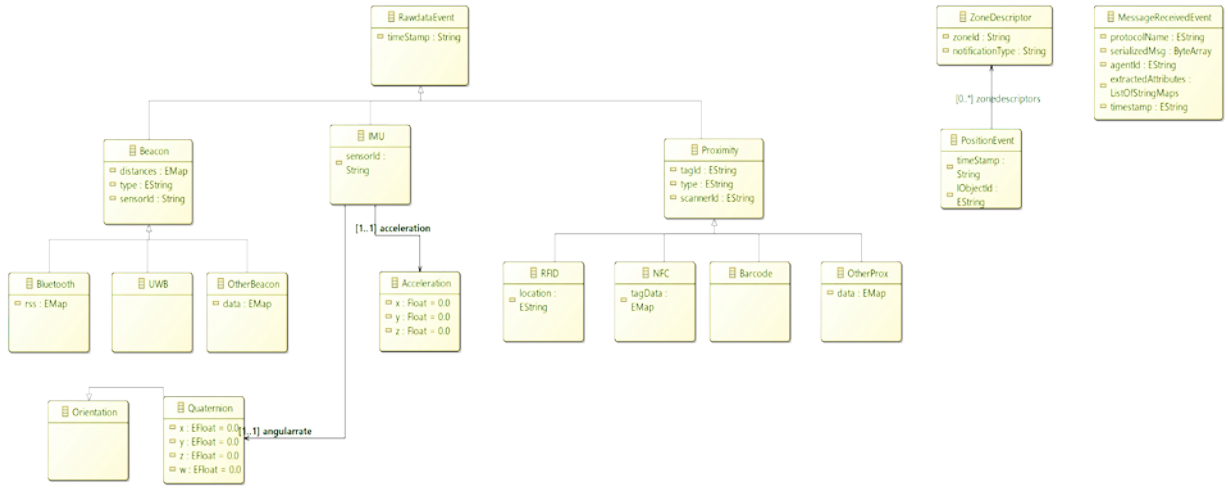

Abbildung 8 Auszug des Packages IPosDevKit des IPos-Datenmodells

Dargestellt sind Datenstrukturen zur Abbildung der ausgetauschten Rohdaten, Positionsdaten und Nachrichten sonstiger Protokolle für die interne Verarbeitung.

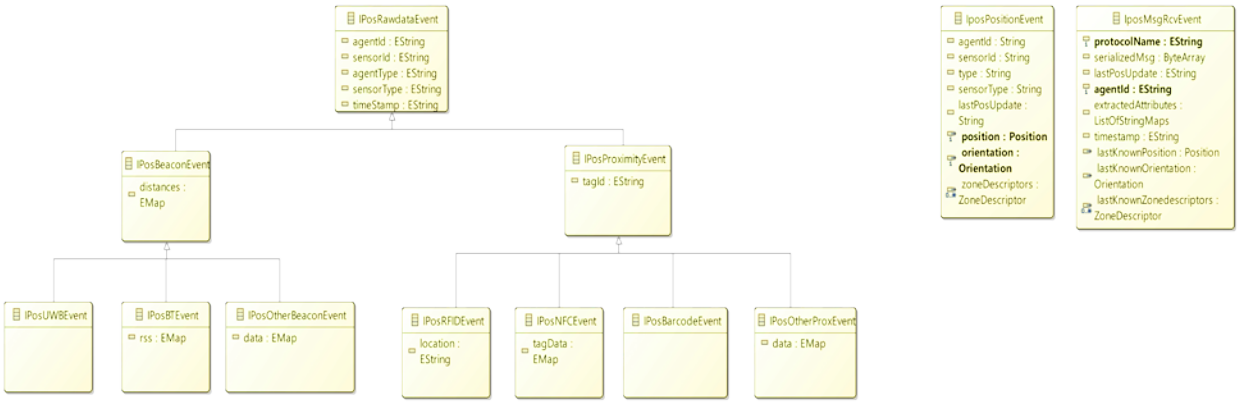

Abbildung 9 Auszug des Packages iPos_Datamodel des IPos-Datenmodells

Dargestellt sind Datenstrukturen zur Abbildung der in der externen Kommunikation ausgetauschten Rohdaten, Positionsdaten und Nachrichten sonstiger Protokolle.

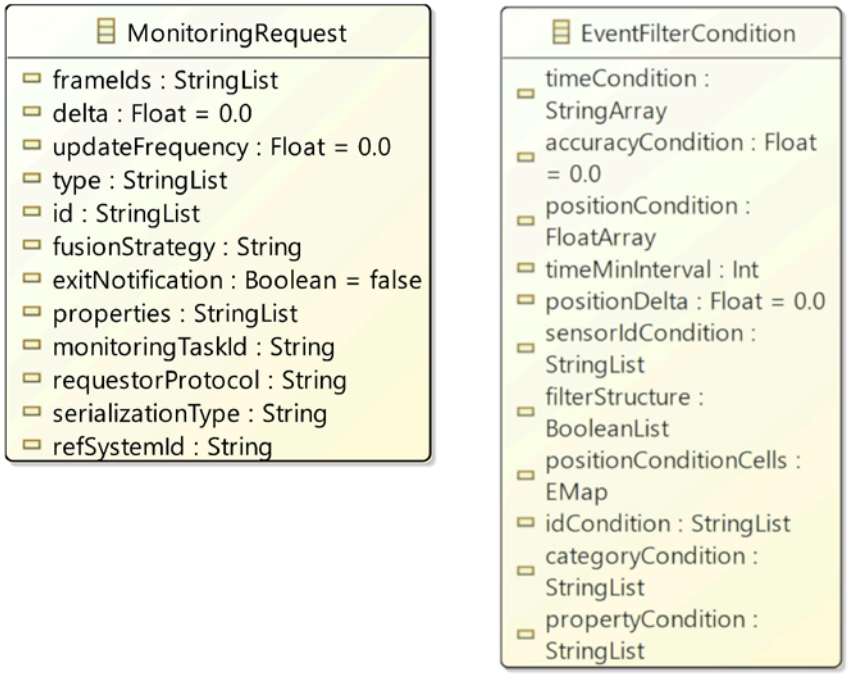

Abbildung 10 Auszug des Packages IPosDevKit des IPos-Datenmodells

Dargestellt sind Datenstrukturen zur Verwendung bei der Systemsteuerung und Konfiguration.

- DataPersistency, MapPersistency

Die Komponenten *DataPersistency* und *MapPersistency* der Entitätsschicht sind für die weitere Arbeit zur Realisierung der Persistenzfunktionalität des IPos-FW vorgesehen. Der Entwurf und die Implementierung dieser Komponenten sind noch ausstehend, dennoch wurden sie in das Architekturdiagramm mit aufgenommen, um die feste Einplanung dieser Funktionalität anzudeuten.

- Datenschicht

In der Datenschicht konnte im Rahmen des Forschungsprojektes die Komponente *IPosWorldModel* realisiert werden. Diese Komponente wird von der Komponente der Entitätsschicht *WorldModelAccess* zur Realisierung der bereitgetellten einheitlichen Zugriffsmöglichkeit auf das Datenmodell des IPos-FW verwendet. Es handelt sich dabei um eine einfache Java-basierte Implementierung des IPos-Datenmodells. Alle Datenstrukturen werden dabei über mit dem EMF-Framework (Eclipse Foundation 2022) generierten Java-Klassen implementiert.

Durchgeführte Arbeiten und Zielerfüllung

Die Arbeiten am Entwurf und der Umsetzung der Komponenten beinhalten die Arbeitspakete 3 und 4. Die Arbeiten wurden in 2 Iterationen durchgeführt. Die erste Iteration beinhaltete eine prototypische Realisierung einer früheren Version der Architektur. Diese Version orientierte sich im Entwurf noch stark an den im Forschungsantrag genannten Komponenten *eXcore, eXsensor, eXhuman, eXdata*. Den Arbeitspaketen lagen die folgenden Ziele zugrunde:

- AP 3: Technisches Konzept für die 4 Software-Komponenten mit Hinblick auf die Gesamtsystemarchitektur aus AP 2; Beschreibung der Sub-Komponenten und deren Interaktion

als Modellbechreibung in Form von DC-, Use-Case- und Sequenzdiagrammen; Mock-Ups (Vorführmodell) für UI; Adaption an Einsatzszenarien / Demonstratoren für Logistik.

- AP 4: Realisierung der 4 Komponenten als Open-Source Software-Elemente mit i) definierten High-Level-Schnittstellen zur Ansteuerung und Integration und ii) Schnittstellenfunktionalität, die zu Umsetzung des Demonstrators benötigt sind.

Im Folgenden wird die Erfüllung dieser Ziele durch den in diesem Kapitel beschriebenen Entwurf diskutiert. Der Entwurf stellt ein technisches Konzept für die Realisierung der 4 im Forschungsantrag beschriebenen Komponenten dar. Alle im Forschungsantrag vorgesehenen Funktionalitäten und Verantwortlichkeiten der 4 Komponenten *eXcore, eXsensor, eXhuman, eXdata* sind im Entwurf enthalten und lassen sich auf Komponenten des Entwurfs abbilden. Die 4 im Forschungsantrag beschriebenen Komponenten stellen die konzeptionelle Grundlage für die im Entwurf enthaltenen Komponenten und deren Organisation dar. Im Einzelnen sind die 4 Komponenten aus dem Forschungsantrag in der folgenden Weise im Entwurf realisiert:

- *eXcore*

  - Die Komponente *eXcore* aus dem Forschungsantag wird im Entwurf durch die Komponenten der Anwendungsfall-Schicht realisiert, insbesondere durch die Komponente *PositionMonitoringController*. Diese Komponente entscheidet über die Weiterverarbeitung empfangener Nachrichten und bindet häufig benötigte Algorithmen ein, darunter Algorithmen zur Positionsberechnung und Koordinatentransformation.

- *eXsensor*

  - Die Komponente *eXsensor* aus dem Forschungsantrag wird im Entwurf in erster Linie durch die Komponente *GenericSensorValueProcessor* realisiert. Diese Komponente akzeptiert das vom IPos-FW erwartete generische Positionssensordatenformat und trägt so zur Integration unterschiedlicher Positionssensortechnologien bei. Darüber hinaus wurden im Projekt in einem Entwicklungswerkzeug einzelne Bausteine für die Plattformen *Android* und *Raspberry Pi* zur einfachen Anbindung von Positionssensoren an das IPos-FW geschaffen. Damit unterstützen auch diese Bausteine Funktionalität der im Forschungsantrag genannten Komponente *eXsensor*. Weiterhin unterstützt die Komponente *PositionMonitoringController* die Funktionalität der Registrierung neuer Positionssensoren und die Komponenten *IPos-Position-Calculation* und *Sensor-Data-Fusion* unterstützen bei der Interpretation der Sensordaten (Umsetzung in Positionen bzw. Umwandlung von empfangenen Beacon-Signalstärken in Distanzen) bzw. bei der Sensordatenfusion *("Merging mit Zusatzinformationen").*

  - Registrierung Sensoren, Berechnung auf Geräten

- *eXhuman*

  - Ein großer Teil der im Forschungsantrag für die Komponente *eXhuman* vorgesehenen Funktionalität wird nicht im IPos-FW selbst implementiert, sondern in den über das IPos-FW zu einem Indoorpositionssystem (IPS) integrierten Einzelkomponenten eines IPS. Im Forschungsprojekt ist mit der *Frontend*-Anwendung eine derartige Komponente entstanden. Diese Komponente könnte gemeinsam mit dem IPos-FW zur Realisierung der im Forschungsantrag für die *eXhuman*-Komponente vorgesehenen Funktionalität eingesetzt

werden. Alternativ könnte aber auch eine andere Komponente zur Realisierung dieser Funktionalität an das IPos-FW angebunden werden. Eine genaue Beschreibung der *Frontend*-Komponente erfolgt im Abschnitt „*Sensordatenfusion*“ zur Fallstudie Sensordatenfusion. Ein Teil der vorgesehenen *eXhuman*-Funktionalität wird durch die Komponente *SimpleSceneIntegration* realisiert. Diese Komponente akzeptiert Nachrichten von anwendungsspezifischen Erweiterungen des IPos-FW zur Anbindung existierender Anwendungen (darunter auch grafische Anwendung zur Nutzer-Interaktion an das IPos-FW.

- *eXdata*

  - Die im Forschungsantrag vorgesehene Komponente *eXdata* wird im Entwurf durch die Komponenten der Entity- und Datenschicht sowie durch die Komponente *AdministrationController* der Anwendungsfallschicht realisiert. Über diese Komponenten können Agenten registriert werden, auf den Datenbestand des IPos-FW gemäß dem definierten Datenmodell zugegriffen werden sowie dieser gepflegt werden.

Mit der Realisierung der im Forschungsantrag für die Komponenten vorgesehenen Funktionalitäten ist die Erfüllung des Zieles „*technisches Konzept für die 4 Software-Komponenten“* (AP3) bzw. das Ziel „*Realisierung der 4 Komponenten als Open-Source Software-Elemente“* (AP4) gezeigt.

Die Struktur und das Verhalten des IPos-FW wurden ebenfalls über UML-Diagramme beschrieben. Als Ergebnisse dieser Arbeit sind in diesem Kapitel ein Sequenzdiagramm zur Beschreibung des Verhaltens der Komponente *PositionMonitoringController* dargestellt sowie ein Klassendiagramm zur Beschreibung des Datenmodells des IPos-FW. Im vorherigen Kapitel zur Architektur des IPos-FW wurde ein UML-Komponentendiagramm dargestellt und ausführlich beschrieben.

Auf die Realisierung verschiedener Fallstudien auf der Grundlage des in diesem Kapitel beschriebenen Entwurfs wird im Kapitel „Fallstudien“ eingegangen. Die Arbeiten an der Integration der in diesem Kapitel beschriebenen Komponenten zu einem Gesamtsystem wurden im Rahmen des Arbeitspaketes 6.2 durchgeführt.

Die Implementierung des in diesem Kapitel beschriebenen Entwurfs kann in einem OpenSource-Projekt ist eingesehen werden: (IPos_GitLab n.d.). Die Bereitstellung der Ergebnisse erfolgte im Rahmen der Bearbeitung des Arbeitspaketes 8.2.

Test

Einzeltest der Funktionalitäten

Eventfilter

Um die Funktionalität des Eventfilters zu testen, wurde eine Testbench entwickelt. Die Testbench initialisiert zunächst einen Eventfilter mit einer vordefinierten Konfigurationsdatei. Dann wird der Eventfilter aufgerufen, um verschiedene Testereignisse zu verarbeiten. Die Ergebnisse werden im Terminal angezeigt.

Die folgende Tabelle gibt eine Übersicht über die durchgeführten Tests. Über jeden Test wird die korrekte Arbeitsweise des Eventfilters für eine bestimmte Filterbedingung getestet. Die Tabelle stellt die Testdaten (linke Spalte) den getesteten Filterbedingungen (rechte Spalte) gegenüber.

Tabelle 2 Testdaten und die getesteten Filterbedingungen

Testdaten	Getestete Konditionen
Zeitstempel in und aus dem definierten Intervall	Zeit
Verschiedene Kategorien	Kategorie
Positionen inner- und außerhalb des definierten Bereichs	Position
Verschiedene Agenten-IDs	Agent ID
Verschiedene Sensortypen/IDs	Sensor ID
Verschiedene Genauigkeiten	Genauigkeit
Eventreihen mit unterschiedlichen Positionen	Position Delta
Eventreihen mit unterschiedlichen Zeitabständen	Zeit des minimalen Aktualisierungsintervalls

SRSConversion

Um die Funktionalität der Komponente *SRSConversion* (Koordinatentransformation) zu testen, wurde ein Prüfstand entwickelt. Die Komponente *SRSConversion* nimmt ein *placing*-Objekt (enthält Position und Orientierung eines Agenten) und ein Bezugssystemobjekt als Eingabe. Der Komponente werden verschiedene *placing*- und Bezugssystemobjekte gegeben, um ihre Funktionalität zu testen. Die folgende Tabelle gibt eine Übersicht der übergebenen Objekte.

Tabelle 3 Test SRSConversion

Eingabe	Getestete Funktionalität
Bezugssystem ohne Drehung	einfache Koordinatenumrechnung
Bezugssystem mit Drehung	Koordinatenumrechnung mit Drehung

Die in diesem Abschnitt vorgestellten Testaktivitäten wurden im Rahmen der Arbeitspakete 5.1, 5.2 und 6.3 durchgeführt.

Fallstudien

In den folgenden Abschnitten werden die im Rahmen des Forschungsprojektes durchgeführten Fallstudien näher beschrieben. In jeder Fallstudie wird die Eignung des IPos-FW für die Entwicklung von Indoor-Positionssystemen (IPS) untersucht. Dabei wird in jeder Fallstudie ein IPS aus bereits vorhandenen Komponenten entwickelt. Das IPos-FW wird dabei als Kernbestandteil des IPS zur Integration der bestehenden Komponenten eingesetzt. Die Fallstudien unterscheiden sich nach den technologischen Besonderheiten der zu integrierenden Komponenten. Integriert werden unterschiedliche Positionssensortechnologien, Komponenten

mit unterschiedlicher Geschäftslogik zur Positionsdatenverarbeitung sowie Komponenten, die eine unterschiedliche Benutzerschnittstelle bereitstellen. Die Fallstudien wurden im Rahmen der Arbeitspakete 7.1 und 7.4 aus dem Forschungsantrag durchgeführt.

Industrierobotik

Szenario

Das Ziel der Fallstudie *Industrierobotik* ist es zu zeigen, dass vorhandene Arbeitssysteme durch das IPos-FW leicht um die Fähigkeit zur Einbeziehung von Umgebungsinformationen in die eigene Arbeit bzw. Entscheidungsfindung erweitert werden können. Zu diesem Zweck haben wir in dieser Fallstudie ein Szenario realisiert, in dem das Arbeitssystem ein Roboterarm ist und seine Aktionen von der Position eines Menschen beeinflusst werden sollen. Der Roboterarm hat die Entscheidung zu treffen, ob ein bereitstehendes Werkstück in den Behälter auf seiner linken Seite- oder in den Behälter auf seiner rechten Seite gelegt werden soll. Die vom Roboterarm zu treffende Entscheidung hängt dabei wie folgt von der Position des Menschen ab: Befindet sich der Mensch links vom Roboterarm, wird das Werkstück in den Behälter auf der rechten Seite des Roboterarms geworfen. Befindet sich der Mensch hingegen rechts vom Roboterarm, wird das Werkstück in den Behälter auf der linken Seite des Roboterarms geworfen. In Abbildung 11 wird das Szenario über ein UML-Aktivitätsdiagramm beschrieben:

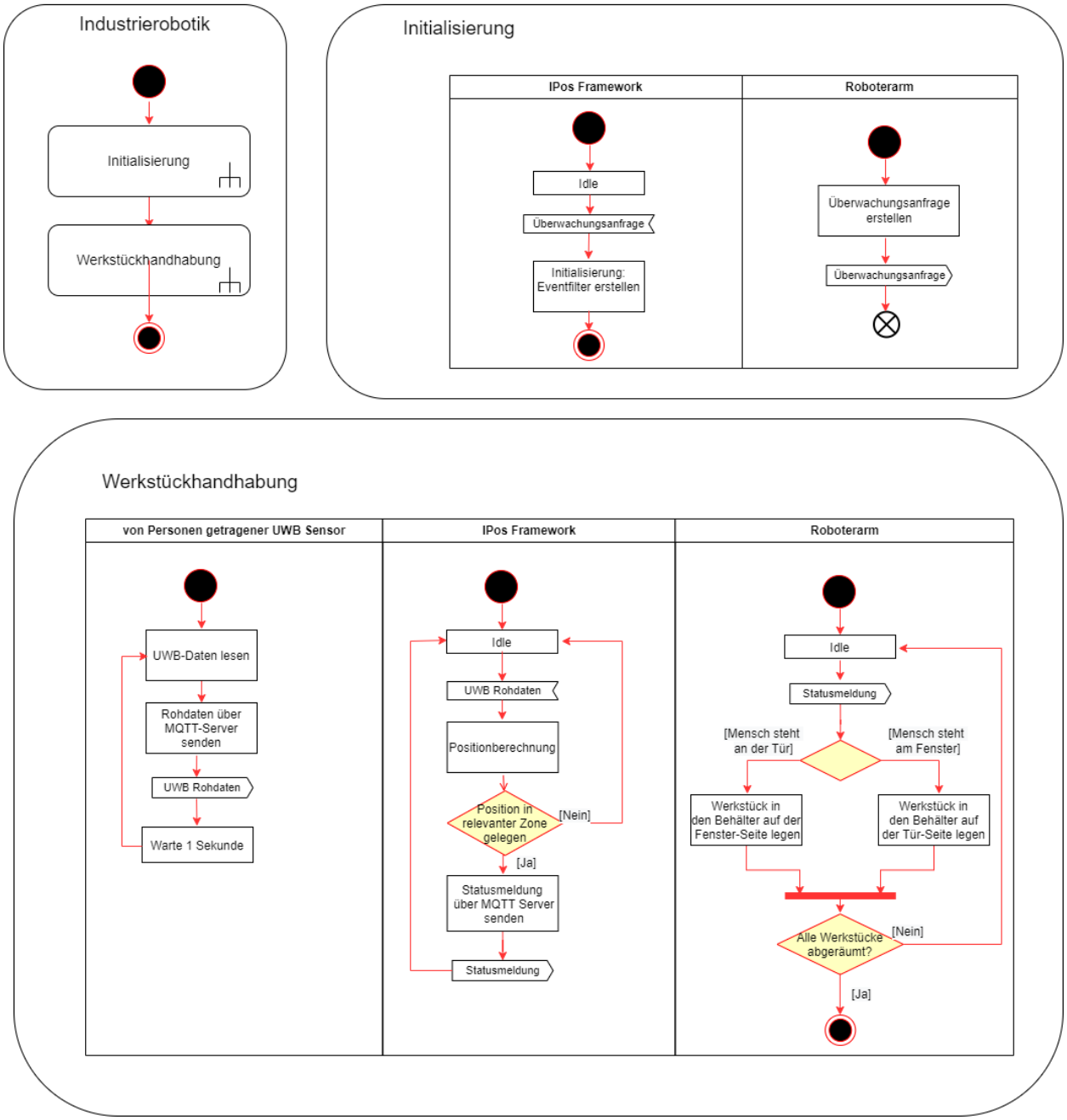

Abbildung 11 Aktivitätsdiagramm zur Fallstudien Industrierobotik

Technische Realisierung

In der Testumgebung sind vier stationäre UWB-Tags installiert. Ein Mensch/AGV trägt einen UWB-Empfänger. Dieser Empfänger empfängt die von allen vier stationären UWB-Tags ausgesendeten Signale und ermittelt die eigene Distanz von jedem der UWB-Tags (*Rohdaten*). Die so ermittelten UWB-Rohdaten werden an das IPos-FW gesendet. Das IPos-FW berechnet dann die Position des Menschen und sendet Positionsaktualisierungen an den Roboterarm. Die Software des Roboterarms ist Bestandssoftware, die zunächst an das vom IPos-FW bereitgestellte und erwartete Datenformat angepasst werden musste. Nachrichten zur Einrichtung von Überwachungsaufträgen mussten an das IPos-FW versendet werden können. Ebenso mussten Positionsnachrichten vom IPos-FW empfangen und verarbeitet werden

können. Der automatisch gesteuerte Roboterarm platziert Lego-Stücke basierend auf der Position des Menschen in bestimmten Boxen.

Um Rohdaten vom UWB-Sensor zu sammeln, wird ein Raspberry Pi so programmiert, dass er die von einem UWB-Tag empfangenen UWB-Rohdaten liest und Nachrichten über einen MQTT-Server an das IPos-FW sendet. Dieses Programm, das auf dem Raspberry Pi läuft, ist modular aufgebaut und kann leicht erweitert werden, um andere Arten von Indoor-Positionssensortechnologien zu unterstützen.


```
{
   "monitoringRequests":[
      {
         "frameIds":[
            "cobot1_door_zone",
            "cobot1_window_zone"
         ],
         "monitoringTaskId":"RobolabMonitoringCeti",
         "refSystemId":"CETI_ROBOTARM_CELL",
         "serializationType":"protobuf"
      },
      {
         "frameIds":[
            "cobot1_door_zone",
            "cobot1_window_zone"
         ],
         "monitoringTaskId":"Industrierobotik",
         "refSystemId":"ROOT",
         "serializationType":"json"
      }
   ]
}
```


Abbildung 12 Beispiel für eine Überwachungsanfrage

Zu Beginn des Experiments sendet das Arbeitssystem (Roboterarm) eine benutzerspezifische Konfigurationsnachricht an das IPos-FW, in der die zu überwachenden Agenten und Zonen definiert werden. Abbildung 12 zeigt ein Beispiel der Nachricht im JSON-Format.

In diesem speziellen Fall sind die zu überwachenden Agenten Menschen: "type" ["HUMAN"] und es werden zwei Bereiche definiert: links vom Roboterarm (*cobot1_window_zone*) und rechts vom Roboterarm (*cobot1_door_zone*). Das Framework startet dann einen Überwachungsprozess, in dem ein Event-Filter entsprechend konfiguriert wird. Im Verlauf des Experiments werden relevante PositionEvents durch den Event-Filter geleitet und an das Arbeitssystem gesendet. Der Roboterarm legt dann die Lego-Stücke in einer Kiste abseits des Menschen ab. Abbildung 13 zeigt eine Momentaufnahme des Experiments.

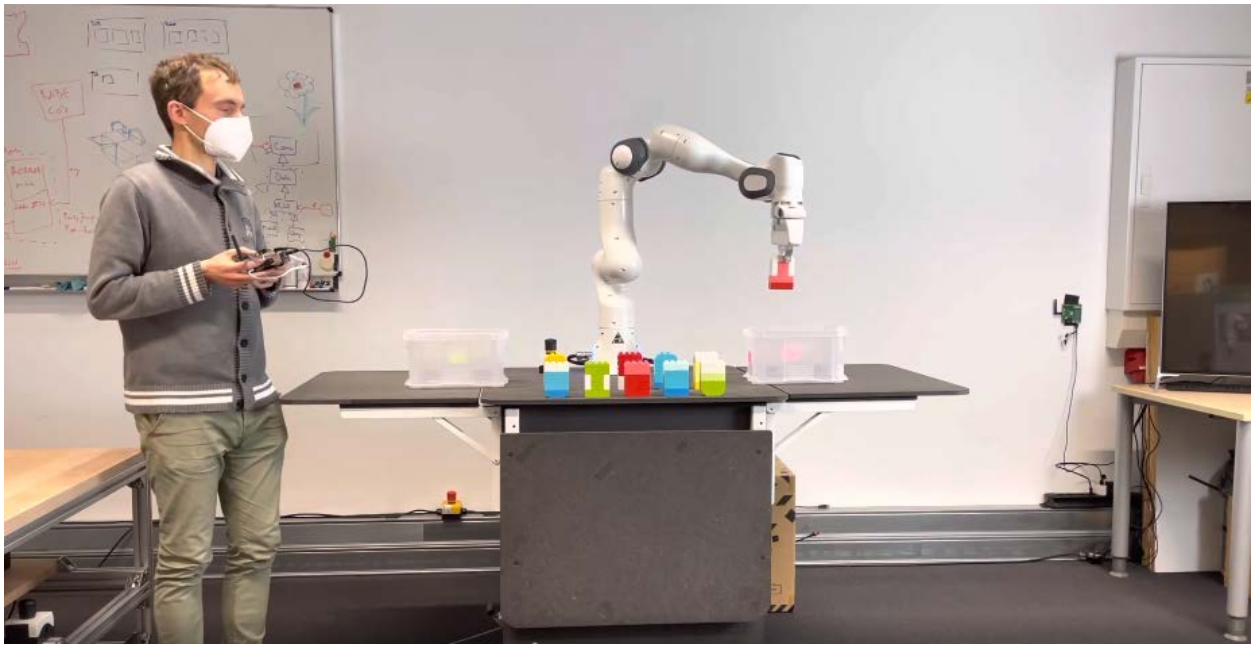

Abbildung 13 Momentaufnahme Industrierobotik

Sensordatenfusion

Szenario

In der Fallstudie *Sensordatenfusion* werden die Fähigkeiten des IPos-FW bezüglich der Ermittlung der Position eines Agenten beim Vorhandensein mehrerer konkurrierender Positionssensoren untersucht. Konkret geht es um die Untersuchung der Praxistauglichkeit des IPos-FW in folgendem Szenario: Ein AGV sei ausgestattet mit einem UWB-Empfänger und einem NFC-Scanner. In der Umgebung des AGV seien 4 UWB-Tags platziert. Ebenso seien NFC-Tags an unterschiedlichen Stellen auf den Boden geklebt. In diesem Szenario besteht für den AGV also sowohl die Möglichkeit, seine aktuelle Position über einen UWB-Positionssensor oder über einen NFC-Positionssensor zu ermitteln. Das unter Verwendung des IPos-FW zu entwickelnde Indoor-Positionssystem stelle die folgenden Funktionalitäten bereit: Beim Vorhandensein mehrerer konkurrierender Positionssensoren für einen Agenten werden nicht alle ermittelten Positionen weitergeleitet, sondern nur die aktuell ermittelte Position diejenigen Positionssensoren mit der höchsten Genauigkeit am aktuellen Aufenthaltsort des AGV. Die aktuelle Position des AGV wird grafisch auf einer Karte dargestellt. Aus der Darstellung geht die Genauigkeit der ermittelten Position hervor. In Abbildung 14 wird das Szenario über ein UML-Aktivitätsdiagramm beschrieben:

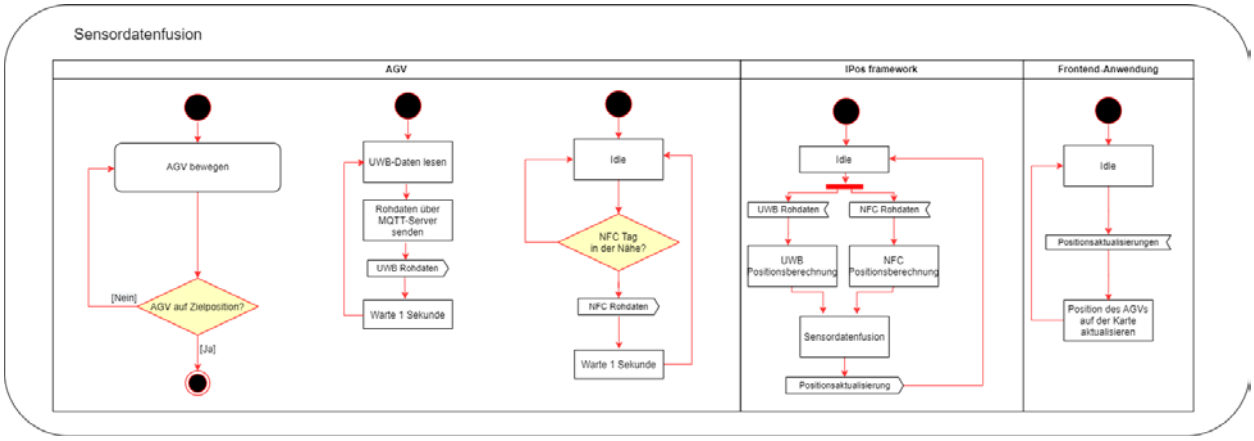

Abbildung 14 Aktivitätsdiagramm zur Fallstudien Sensordatenfusion

Technische Realisierung

Zur technischen Realisierung des beschriebenen Szenarios wurde ein Turtlebot als AGV verwendet. Dieser fahrbare Roboter unterstützt die Informationsverarbeitung über einen Raspberry Pi. Genauere Informationen zur technischen Ausstattung des Turtlebots finden sich auf der Webseite [Turtlebot].

Der Turtlebot wurde mit zwei verschiedenen Positionssensoren ausgestattet. Zum einen wurde der Raspberry Pi zu einem UWB-Positionssensor erweitert. Dazu wurde der Raspberry Pi um einen Chip zum Empfang von UWB-Nachrichten ergänzt und vier UWB-Tags in der Umgebung des Turtlebot platziert. Außerdem wurde die Software des Raspberry Pi um die Möglichkeit der Verarbeitung der empfangenen UWB-Signale ergänzt. Über die Software erfolgt dabei die Ermittlung der Distanzen und das Erstellen und Versenden der Distanzen als UWB-Rohdaten an das IPos-FW.

Zum anderen wurde der Turtlebot mit einem NFC-Positionssensor ausgerüstet. Dieser besteht aus einem NFC-Scanner und einem WLAN-Chip. Der NFC-Positionssensor wurde für den MQTT-Versand der NFC-Daten im vom IPos-FW erwarteten Datenformat programmiert. Informationen zum verwendeten WLAN-board finden sich auf der Webseite [LOLIN-D1-Mini]. Der so ausgerüstete Turtlebot ist in der Lage, beim Überfahren von den auf den Boden aufgeklebten NFC-Tags eine Nachricht an das IPos-FW zu senden.

Das IPos-FW wurde zur Verarbeitung der vom Turtlebot empfangenen UWB- bzw. NFC-Posititionssensorsignale eingesetzt. Die benötigte Funktionalität der selektiven Weiterleitung von empfangenen Positionen wurde dabei von der Komponente *Sensor-Data-Fusion* realisiert (siehe Abschnitt „Entwurf und Umsetzung der Komponenten“). Die Weiterleitung erfolgt nach der Genauigkeit der empfangenen Positionen und nach ihrem Alter. Vor zu langer Zeit bekannt gewordene Positionen werden aufgrund fehlender Aktualität bei der Filterung nicht mehr berücksichtigt. Bei korrektem Verhalten des IPos-FW ergibt sich das folgende Verhalten: Solange sich der Turtlebot über keinem NFC-Tag befindet, wird die (an jedem Ort berechenbare) Position des UWB-Positionssensoren weitergeleitet. Wenn sich der Turtlebot über einem NFC-Tag befindet, wird die Position des NFC-Tags weitergeleitet. Nachdem der Turtlebot das NFC-Tag verlassen hat, wird noch für einen kurzen Zeitraum die Weiterleitung der vom UWB-Positionssensor gelieferten Position unterbunden (die vom NFC-Positionssensor gelieferte Position ist noch aktuell). Danach wird wieder die vom UWB-Positionssensor ermittelte Position weitergeleitet.

Vor seiner Verwendung zur Realisierung der beschriebenen Verarbeitung von Signalen mehrerer konkurrierender Positionssensoren muss das IPos-FW vom Anwender konfiguriert werden. Dem IPos-FW sind dabei folgende Informationen mitzuteilen:

- Auswahl des Algorithmus zur Sensordatenfusion
- Definition der Standorte der NFC-Tags als Referenzpunkte
- Registrierung der verschiedenen Positionssensoren

Die Registrierung der verschiedenen Positionssensoren erlaubt es dem IPos-FW zu erkennen, welche Positionssensoren welchem Agenten zugeordnet sind. Nur dadurch kann das Vorhandensein konkurrierender Positionssensoren auf einem Agenten erkannt werden und die vorhandene Funktionalität zur Sensordatenfusion angewandt werden.

Zur Realisierung des beschriebenen Szenarios wird eine grafische Darstellung der aktuellen AGV-Positionen benötigt. Zu diesem Zweck wurde eine eigene Anwendung entwickelt und an das IPos-FW angebunden. Die Anwendung bezieht das benötigte *OpenStreetMap*-Kartenmaterial über die Bibliothek *Leaflet*. Eine genauere Besprechung dieser Anwendung erfolgt zur Fallstudie *VDA5050*. Die höhere Genauigkeit der NFC-Positionssensoren wird durch die geringe Größe des Kreises in der Umgebung der AGV-Position angezeigt.

Orderpicker

Szenario

In der *Orderpicker*-Fallstudie wird versucht, auf Grundlage des IPos-FWs ein Indoor-Positionssystem mit einer relativ komplexen Geschäftslogik zu entwickeln. Der Fallstudie liegt folgendes Szenario zugrunde: Benötigt werde ein Indoor-Positionssystem, welches die Fehlererkennung bei der Kommissionierung unterstützt. Das Indoor-Positionssystem erlaube die Bestimmung der Position der Hand des Kommissionierers mit einer ausreichenden Genauigkeit. Die Genauigkeit soll ausreichen, um erkennen zu können, in welchen Behälter der Kommissionierer gegriffen hat. Die Positionsbestimmung sei jederzeit und an jedem Ort in der Lagerhalle möglich. Darauf aufbauend erlaube das Indoor-Positionssystem die Definition eines Kommissionierauftrages. Aus diesem Kommissionierauftrag soll die Reihenfolge der vom Kommissionierer zu pickenden Artikel eindeutig hervorgehen. Der Kommissionierauftrag werde dem Indoor-Positionssystem gemäß dem Datenmodell des OFBiz-Enterprise Ressource Planning Systems [OFBiz] zur Verfügung gestellt. Das Indoor-Positionssystem sei in der Lage, aus den OFBiz-Daten einen Kommissionierauftrag abzuleiten und zu verarbeiten. Das Indoor-Positionssystem nutze den vorhanden Kommissionierauftrag für die Fehlererkennung. Dabei werden insbesondere die folgenden Funktionalitäten verlangt:

- Der Kommissionierauftrag, d. h. das SOLL-Verhalten des Kommissionierers, werden angezeigt
- Mit Fortschreiten des Kommissioniervorganges werde das IST-Verhalten aktualisiert. Es muss dabei ersichtlich werden, in welcher Reihenfolge aus den Behältern Artikel tatsächlich herausgenommen bzw. hineingelegt wurden.
- Abweichungen des IST-Verhaltens vom SOLL-Verhalten sind deutlich zu kennzeichnen
- Die Anzeige erfolgt über eine intuitive grafische Benutzeroberfläche

In Abbildung 15 wird das Szenario über ein UML-Aktivitätsdiagramm beschrieben:

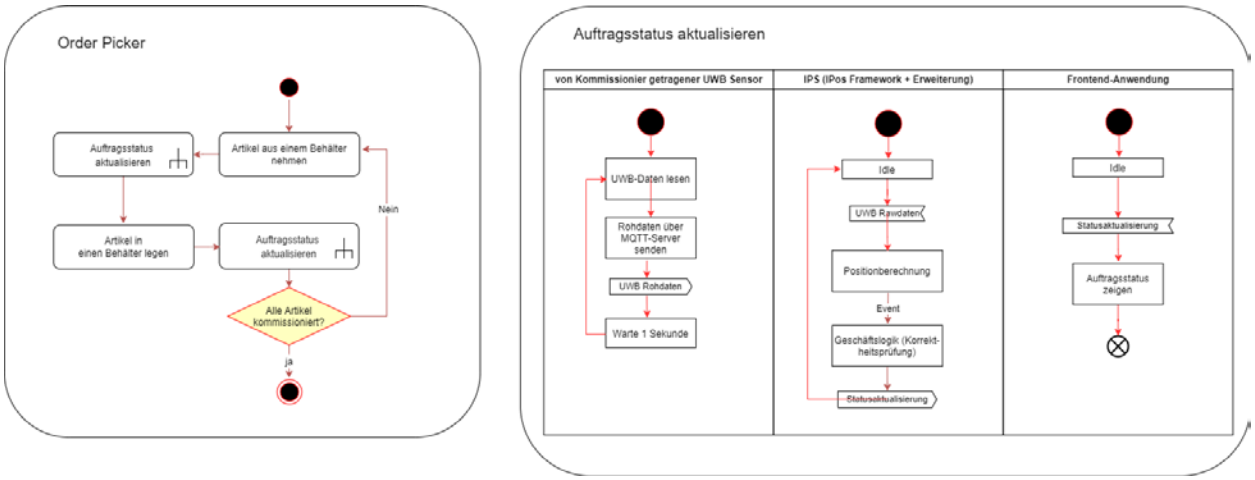

Abbildung 15 Aktivitätsdiagramm zur Fallstudien Order Picker

Technische Realisierung

Zur technischen Realisierung des beschriebenen Szenarios wurde als Grundlage das IPos-FW eingesetzt sowie der bereits zur Fallstudie *Industrierobotik* beschriebene UWB-Positionssensor. Dieser Raspberry Pi-basierte Positionssensor kann aufgrund seiner geringen Größe in der Hand getragen werden, sodass er sich für die Durchführung dieser Fallstudie eignet. Das beschriebene Szenario setzt die Verwendung des Enterprise Ressource Planning (ERP) Systems *OFBiz* voraus. Dieses ERP-System wird daher als existierende Software aufgefasst und in der Schichtenarchitektur des IPS für diese Fallstudie mitberücksichtigt. Daher enthält diese Schichtenarchitektur in der Anwendungsschicht die Komponente *OFBiz* im Abschnitt Orderpicker zur Architektur der Lösung). Zur technischen Realisierung der benötigten Funktionalität als Erweiterung des IPos-FW zum gewünschten Indoor-Positionssystem wurde in der Anwendungsschicht eine weitere Komponente namens *Orderpicker-Integration* entwickelt. Diese Komponente ist für die technische Realisierung der folgenden Funktionalitäten verantwortlich:

- Implementierung der benötigten Geschäftslogik
- Überführung des vorausgesetzten OFBiz-Datenmodells in ein IPS-spezifisches Datenmodell zur Erfassung von Kommissionieraufträgen
- Anbindung der Erweiterungen an das IPos-FW, d. h. Durchführung der benötigten Konfiguration des IPos-FW und Akzeptanz und Verarbeitung der vom IPos-FW empfangenen Positionen

Im Folgenden wird die technische Realisierung der einzelnen Funktionalitäten beschrieben.

Implementierung der benötigten Geschäftslogik

Zur Implementierung der benötigten Geschäftslogik wurden mehrere Klassen des *devkits* verwendet. Die Geschäftslogik beinhaltet die Überprüfung der korrekten Reihenfolge von eintreffenden Positionsereignissen. Die Klasse *ZoneSequenceRule* des *devkits* unterstützt derartige Anwendungsfälle und konnte daher für die Implementierung der Geschäftslogik verwendet werden. Zur Anpassung dieser Klasse an diese Fallstudie wurde sie erweitert (konkrete Klasse: *SeqPicklistRule*). Die Klasse erzeugt eine Sequenz von Filterstufen, wobei

jeder Filterstufe eine Zone zugeordnet ist. Die vom IPos-FW erhaltenen Positionsnachrichten enthalten die aktuelle Position des Agenten (in dieser Fallstudie ist der Agent der Kommissionierer) sowie die Namen der Zonen, zu denen diese Position gehört. Anhand dieser Informationen kann die Zonenreihenfolge und damit die Reihenfolge ermittelt werden, in der der Kommissionierer auf die Behälter zugegriffen hat. Außerdem kann die Korrektheit dieser Reihenfolge überprüft werden. Fehler können als Abweichungen von der erwarteten Zonenreihenfolge erkannt werden.

Die Geschäftslogik umfasst auch die grafische Darstellung des Fortschritts des Kommissionierers bei der Abarbeitung des Kommissionierauftrages sowie die Darstellung erkannter Fehler. Zur Realisierung dieser Funktionalität wurde eine eigene Web-Anwendung entwickelt und mit der Komponente *Orderpicker-Integration* verbunden (Kommunikationsprotokoll: MQTT). Die grafische Darstellung zeigt eine Tabelle, in der die SOLL-Behälter gemäß des Kommissionierauftrages eingetragen sind. Diesen SOLL-Behältern werden die erkannten IST-Behälter gegenübergestellt. Mit Fortschreiten des Kommissioniervorganges wird die Spalte mit den IST-Behältern gefällt. Abweichungen von der Spalte der SOLL-Behälter werden farblich gekennzeichnet (Abweichung: rot, Übereinstimmung: grün). Abbildung 16 zeigt die für diese Fallstudie entwickelte grafische Benutzeroberfläche.

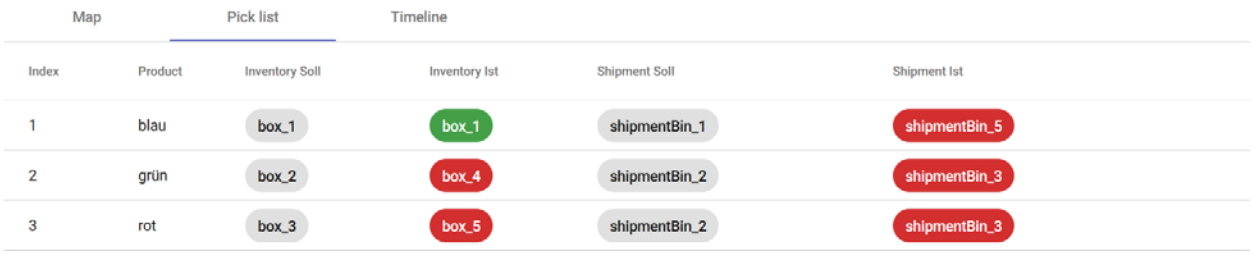

Abbildung 16 grafische Benutzeroberfläche für Fallstudie Orderpicker

Überführung des vorausgesetzten OFBiz-Datenmodells

Die vom Kommissionierer abzuarbeitende Stückliste wird der Erweiterung des IPos-FW über Datensätze zur Verfügung gestellt, deren Struktur den Vorgaben des OFBiz-Datenmodelles entspricht (siehe (OFBiz n.d.)). Der relevante Ausschnitt des OFBiz-Datenmodelles ist in Abbildung 17 dargestellt.

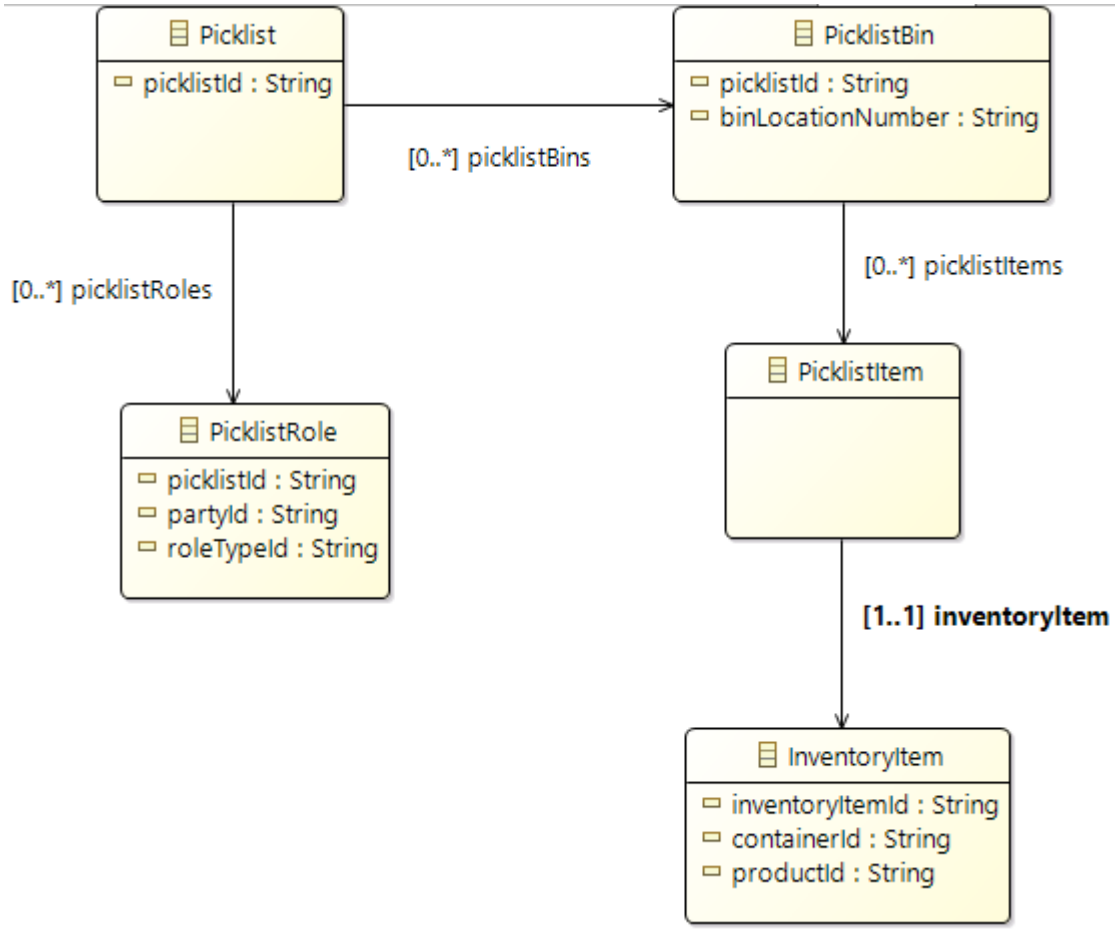

Abbildung 17 Ausschnitt OFBiz-Datenmodell

In einem realen Einsatzszenario könnten diese Daten vom ERP-System *OFBiz* als im Unternehmen vorhandene Bestandssoftware bereitgestellt werden. Die Erweiterung des IPos-FW akzeptiert diese Daten, erzeugt daraus eine Filterkonfiguration und startet damit einen Ereignisfilter der Positionsereignisse in einer Reihenfolge und mit einer Zonenzuordnung erwartet, die auf eine korrekte Abarbeitung der zuvor erhaltenen Stückliste durch den Kommissionierer schließen lässt.

Anbindung der Erweiterungen an das IPos-FW

Zur Realisierung der technischen Anbindung der Erweiterungen an das IPos-FW wurde intensiv mit den vom *devkit* zur Verfügung gestellten Klassen gearbeitet. Realisiert wurde die initiale Konfiguration des IPos-FW für das dieser Fallstudie zu Grunde liegende Szenario. Dabei wurden mithilfe des devkits Nachrichten an das IPos-FW gesendet, die die folgenden Änderungen an den Konfigurationseinstellungen bewirkten:

- Registrierung von zusätzlichen Zonen. Dabei wurde für jeden Behälter, aus dem der Kommissionierer einen Artikel herausnehmen bzw. hereinlegen kann, eine eigene Zone definiert.
- Einrichtung eines Überwachungsauftrages. Der Überwachungsauftrag definiert die zu überwachenden Zonen sowie den Agenten, dessen Position zu überwachen ist.

Bei der Akzeptanz der vom IPos-FW erhaltenen Positionsnachrichten wurde auf die Transformator-Klassen des *devkits* zurückgegriffen.

VDA5050

Szenario

In der Fallstudie *VDA5050* wird die Eignung des IPos-FW für die Einbindung von Agenten, welche sich gemäß etablierter, standardisierter Protokolle verhalten. Ausgewählt wurde dafür das Protokoll *VDA 5050* (VDA 2021), welches eine standardisierte Schnittstelle für Kommunikation zwischen AGVs und AGV-Steuerung (Leitstelle) beschreibt. Der Einsatzzweck des IPos-FW besteht in der Herstellung der Interoperabilität zwischen VDA 5050-kompatiblen AGVs und VDA 5050-inkompatiblen Steuerungen bzw. zwischen VDA 5050-inkompatiblen AGVs und VDA 5050-kompatiblen Steuerungen. In dieser Fallstudie wird die grundsätzliche Eignung des IPos-FW für die Unterstützung des Protokolls VDA 5050 untersucht. Dazu wird eine Erweiterung des IPos-FW für die Unterstützung eines Ausschnittes des Protokolls entwickelt. Im Folgenden wird näher auf das Protokoll und den zu unterstützenden Ausschnitt eingegangen.

VDA 5050 ist eine standardisierte Schnittstelle für die AGV-Kommunikation. Konkret betrifft dieser Standard die Kommunikation zwischen AGVs (in Deutschland oft als Fahrerlose Transportsysteme/Transportfahrzeuge (FTS) bezeichnet) und einer übergeordneten Steuerung. Seit 2017 erarbeitet ein gemeinsames Projektteam aus AGV-Nutzern und AGV-Herstellern den Standard VDA 5050 – im Sommer 2020 erschien dessen Version 1.1. Die Schnittstelle ermöglicht eine hardwareunabhängige Kommunikation zwischen Leitstelle und Fahrzeugen. Dies erhöht die Interoperabilität zwischen AGVs und Leitstellen unterschiedlicher Hersteller und damit die mögliche Heterogenität bei der Zusammensetzung der Fahrzeugflotte und der technischen Infrastruktur zur Steuerung des Transportprozesses. Für die Anwender hat mehr Heterogenität viele Vorteile: Zum Beispiel werden große finanzielle Einsparungen möglich; außerdem ergibt sich eine nie da gewesene Flexibilität. Weitere Vorteile werden durch Vereinfachung und Vereinheitlichung erreicht – Effizienz und Qualität steigen.

In dieser Fallstudie wurde ein Ausschnitt des VDA 5050-Standards zur Unterstützung durch das IPos-FW ausgewählt. Dieser Ausschnitt definiert, wann und wie Statusmeldungen einschließlich ihrer Positionsinformationen von AGVs gesendet werden sollen. In vielen Fällen ist das AGV-System ein wichtiger Bestandteil des Intralogistiksystems. Auch deshalb wurde dieser Standard für die Verwendung in dieser Fallstudie ausgewählt, um die Offenheit des IPos-FW für diesen Standard und dessen grundsätzliche Erweiterbarkeit zu zeigen. Die für diese Fallstudie durchgeführten Arbeiten wurden im Rahmen des Arbeitspaketes 7.3 aus dem Forschungsantrag durchgeführt.

Technische Realisierung

Zur technischen Realisierung des beschriebenen Szenarios wurde das IPos-FW um eine Komponente für den Empfang und die Prüfung der VDA 5050-Kompatibilität von Nachrichten erweitert. Der für dieses Szenario vorausgesetzte technische Prozess wurde durch einen VDA 5050-Simulator realisiert, welcher aufbauend auf einem bestehenden AGV-Simulator entwickelt wurde.

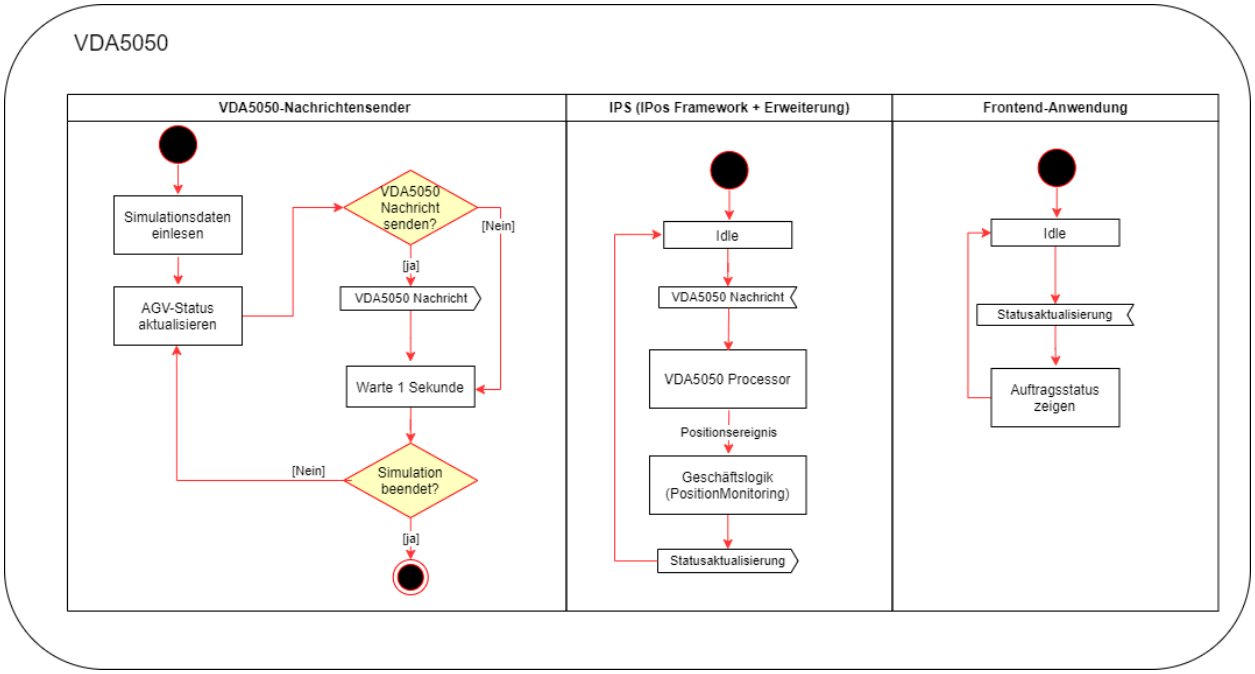

Abbildung 18 Aktivitätsdiagramm zur Fallstudien VDA5050

Der VDA5050-Simulator wurde entwickelt, um Statusaktualisierungen von AGVs gemäß dem VDA 5050-Standard zu generieren. Der VDA 5050-Simulator besteht aus zwei Komponenten: einem AGV-Simulator, einem VDA 5050-Nachrichtensender.

AGV Simulator

Ein bestehender Python-basierter AGV-Simulator wurde erweitert, um die Bewegung mehrerer AGVs zu simulieren. Zu Beginn der Simulation wird das Umgebungssetup geladen, einschließlich der Fahrzeug- und Knotendefinition. Die zu bearbeitenden Aufträge können zu Beginn der Simulation geladen oder während der Simulation dynamisch generiert werden. Dann werden die Bewegungen der Fahrzeuge mit der Python-Bibliothek *simpy* simuliert. Der Status von AGVs während der Simulation, z.B. Position und Geschwindigkeit werden in einer Protokolldatei gespeichert.

Abbildung 19 Visualisierung des AGV-Simulators

AGV Nachrichtensender

Der VDA 5050-Nachrichtensender liest zuerst die Protokolldatei ein und entscheidet dann, wann eine VDA 5050-Statusaktualisierung gemäß dem VDA5050-Protokoll gesendet wird.

Der Status in der Protokolldatei wird dann in eine Statusmeldung gemäß dem VDA 5050-Datenformat übersetzt. Eine Beispielnachricht ist unten dargestellt. Ein Visualisierungstool bietet eine intuitive Darstellung des AGV-Status in Echtzeit (siehe Abbildung 19).

{"headerId": "5", "timeStamp": "2022-05-13T09:44:18.430334+00:00", "manufacturer": "TL_TUD", "serialNumber": "5", "lastNodeId": "0", "batteryState": {"batteryCharge": "100"}, "errors": [{"errorType": "0", "errorLevel": "WARNING"}], "agvPosition": {"x": 11.5042, "y": 6.2589999999999995, "theta": 2.088578185407573}, "loads": [{"loadId": "5"}]}

Schließlich werden die Statusmeldungen mehrerer AGVs zum richtigen Zeitpunkt an einen MQTT-Server gesendet.

Das IPos-FW erhält VDA5050-Statusaktualisierungen von den Fahrzeugen und wandelt sie in Ereignisse im internen Datenmodell um. Für diese Funktionalität war eine Erweiterung des IPos-FW notwendig. Die Unterstützung etablierter Protokolle wurde dabei als Kernfunktionalität eines IPS betrachtet, sodass die notwendige Erweiterung nicht in der Anwendungsschicht, sondern in der Integrationsschicht realisiert wurde. Die Teilschicht *DataModelIntegration* wurde um eine Komponente *VDA5050Processor* erweitert. Diese Komponente nutzt das Kommunikationsprotokoll MQTT und akzeptiert VDA5050-kompatible Nachrichten im JSON-Format. Für die Weiterleitung von Nachrichten, die in einem unterstützten Protokoll (wie *VDA*

5050) geschrieben sind, bietet das IPos-FW zwei Varianten an. Zum einen ist es möglich, einen Teil der enthaltenen Informationen aus der empfangenen Nachricht zu extrahieren und in eine IPos-kompatible Nachricht umzuwandeln, z. B. ein Positionsereignis. Die Komponente *VDA5050Processor* unterstützt diese Variante für das Protokoll *VDA 5050*. Zum anderen ist es möglich, die empfangene Nachricht unverändert in eine IPos-Nachricht *IposMessageReceivedEvent* zu verpacken. Diese Variante bietet den Vorteil, dass auch Informationen, die vom IPos-FW aus der ursprünglich empfangenen Nachricht nicht extrahiert werden konnten, an den Empfänger weitergeleitet werden können. Diese zweite Variante wird von der Komponente *PositionMonitoringController* implementiert. Dank der beschriebenen Funktionalitäten ist es den Anwendern des IPos-FW möglich, in Reaktion auf einen gestellten Überwachungsauftrag die Position eines *VDA 5050*-kompatiblen AGV gemeldet zu bekommen.

Im Rahmen dieser Fallstudie wurde auf der Grundlage des IPos-FW ein Indoor-Positionssystem entwickelt, welches zum einen in der Lage ist, die Position von *VDA 5050*-kompatiblen AGVs zu ermitteln und zum anderen in der Lage ist, diese über eine intuitive grafische Benutzeroberfläche darzustellen. Dazu wurde:

1. das IPos-FW wie oben beschrieben erweitert,

2. eine geeignete grafische Benutzeroberfläche mit Webtechnologien implementiert und

3. eine Integrationskomponente zur Anbindung der erstellten Webanwendung an das IPos-FW in die Anwendungsschicht des IPS aufgenommen.

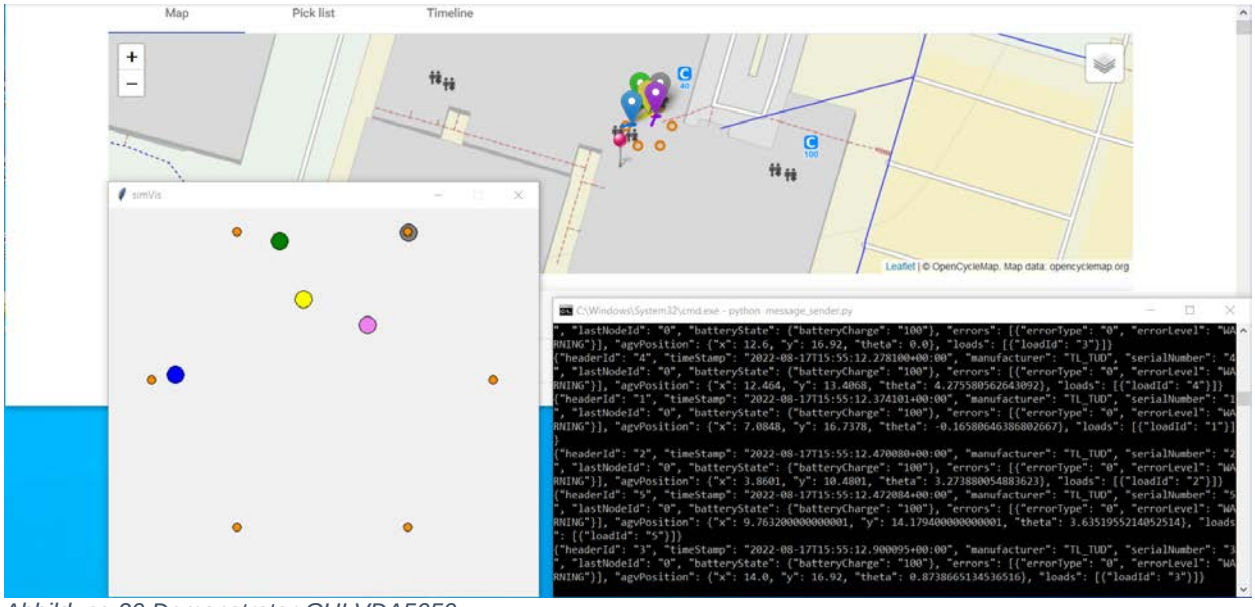

Abbildung 20 Demonstrator-GUI VDA5050

Die Integrationskomponente konnte unter umfangreichem Einsatz der Klassen des *devkits* implementiert werden.

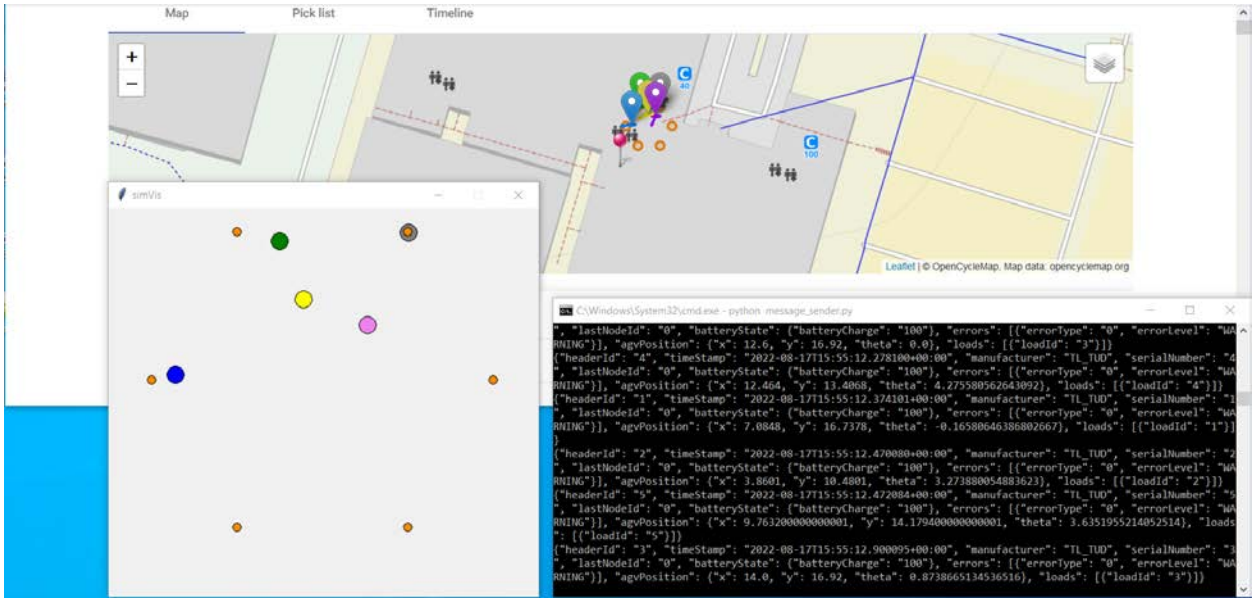

Abbildung 20 zeigt die vollständige Bildschirmansicht bei Ausführung aller zu dieser Fallstudie gehörenden Programme. Unten rechts ist die grafische Ausgabe des entwickelten VDA5050-Simulatoren zu sehen. Dieses Fenster zeigt den *IST-Zustand* der AGVs an. Die von der Webanwendung (Fenster oben links) dargestellte Anordnung der AGVs *soll* diesem Zustand entsprechen. In der dargestellten Abbildung ist dies der Fall. Daraus geht hervor, dass das IPos-FW die zur Beschreibung des *IST-Zustandes* empfangenen *VDA 5050*-kompatible Nachrichten korrekt in IPos-kompatible Nachrichten umwandeln. Das Fenster im unteren Teil des Bildschirms zeigt eine Konsolenansicht der vom IPos-FW empfangenen *VDA 5050*-Nachrichten. Aus diesem Fenster geht hervor, dass das IPos-FW tatsächlich mit *VDA 5050*-kompatiblen Nachrichten beaufschlagt wird.

Tooz

Szenario

In dieser Fallstudie wird der Anpassungsaufwand für die Auslagerung der Positionssensordatenverarbeitung bestehender Software auf das IPos-FW untersucht. Das IPos-FW wird dabei in folgendem Szenario eingesetzt: Der Endnutzer trägt eine Datenbrille und bekommt kontextabhängig bestimmte Informationen über die Datenbrille eingeblendet. Im Hintergrund arbeitet eine zentrale Recheneinheit, deren Aufgabe darin besteht, kontextabhängig zu entscheiden, welche Informationen auf der Datenbrille eingeblendet werden soll. Die Kontextabhängigkeit besteht dabei in diesem Szenario in der Abhängigkeit von der räumlichen Nähe des Endnutzers zu bestimmten Referenzpunkten. Sinnvolle konkrete Einsatzmöglichkeiten für dieses allgemeine Szenario wäre die Unterstützung der ärztlichen Visite in einem Krankenhaus oder die Unterstützung logistischer Prozesse in einem Warenlager durch Warnmeldungen oder sonstige Hinweise an die dort tätigen Mitarbeiter.

Technische Realisierung

Die Hardware des vorhandenen Systems besteht aus einer Datenbrille (Hersteller: Tooz), einem Smartphone (Betriebssystem: Android) und einer zentralen Recheneinheit (nachfolgend genannt *backend*). Die Bestandssoftware besteht aus einer Komponente, die auf dem Smartphone ausgeführt wird, und einer Komponente, die auf dem backend ausgeführt wird. Diese Komponenten haben die folgenden Verantwortlichkeiten:

Smartphone

- Entgegennahme der einzublenden Informationen vom backend
- Weiterleitung der vom backend erhaltenen und für die Einblendung vorgesehenen Informationen an die Datenbrille

Backend

- Entgegennahme der bereitgestellten Kontextinformationen
- Ermittlung der von der Datenbrille einzublenden Informationen unter Verwendung von vorhandenem Wissen (gespeichert in einer Datenbank o. ä.) über weitere Eigenschaften der im Kontext vorhandenen Objekte.
- Versenden der von der Datenbrille einzublenden Informationen an das Smartphone

IPos-FW

In dieser Fallstudie sollen die folgenden Verantwortlichkeiten des Smartphones auf das IPos-FW bzw. auf vom IPos-Projekt bereitgestellte Komponenten und Softwarebausteine (das *devkit*) ausgelagert werden:

- Entgegennahme von Positionssensorsignalen unterschiedlicher Positionssensortechnologien
- Verarbeitung von Positionssensorsignalen
    - Ermittlung des nächstgelegenen Referenzpunktes
    - Ermittlung der dem Referenzpunkt zugeordneten Id (z. B. Id eines Patienten im Krankenhaus oder Id eines Fahrroboters)
- Weiterleitung der Id an das backend

Zur technischen Realisierung dieser Aufgabe wurde in einem ersten Schritt die Bestandssoftware auf dem Smartphone für die Weiterleitung der empfangenen Positionssensorsignale an das IPos-FW angepasst. Die Erstellung wurde implementiert unter Einsatz von *devkit* des IPos-Projektes bereitgestellten Softwarebausteinen. Diese Bausteine erlauben das einfache Erstellen von Nachrichten in dem vom IPos-FW erwarteten Format und den Versand dieser Nachrichten unter Verwendung des Kommunikationsprotokolls MQTT. Für die zukünftige Arbeit könnte in einem zweiten Schritt eine eigenständige Smartphone-App für die Auslagerung der Entgegennahme von den Positionssensorsignalen unterschiedlicher Positionssensorsignale und auch die Weiterleitung dieser Informationen an das IPos-FW entwickelt werden und an die Anwender des IPos-FW bereitgestellt werden.

Zur Implementierung der übrigen Verantwortlichkeiten wurde das IPos-FW in ähnlicher Weise wie in der *Orderpicker*-Fallstudie (siehe Abschnitt „*Orderpicker*“) um die Unterstützung eines sonstigen Datenmodells erweitert. Das so zu einem IPS erweiterte IPos-FW kann den erhaltenen Referenzpunkten eine Id zuordnen. Weiterhin wurde über die Erweiterung dem IPS Unterstützung für das STOMP-Kommunikationsprotokoll hinzugefügt. Dieser Schritt war nötig, um dem IPS die Kommunikation mit dem Backend zu ermöglichen.

Bewertung

Anwenderumfrage

Zur Sicherstellung der Praxisrelevanz der vom IPos-FW unterstützen Technologien wurde unter den Mitgliedern des projektbegleitenden Ausschusses eine Umfrage durchgeführt. In dieser Umfrage wurden Fragen zu Funktionalitäten von Indoor-Positionssystemen sowie zur Verwendung spezieller Technologien thematisiert. Im Folgenden werden über Balkendiagramme die einzelnen Fragen mit einem Überblick über die Antworthäufigkeiten der Umfrageteilnehmer vorgestellt.

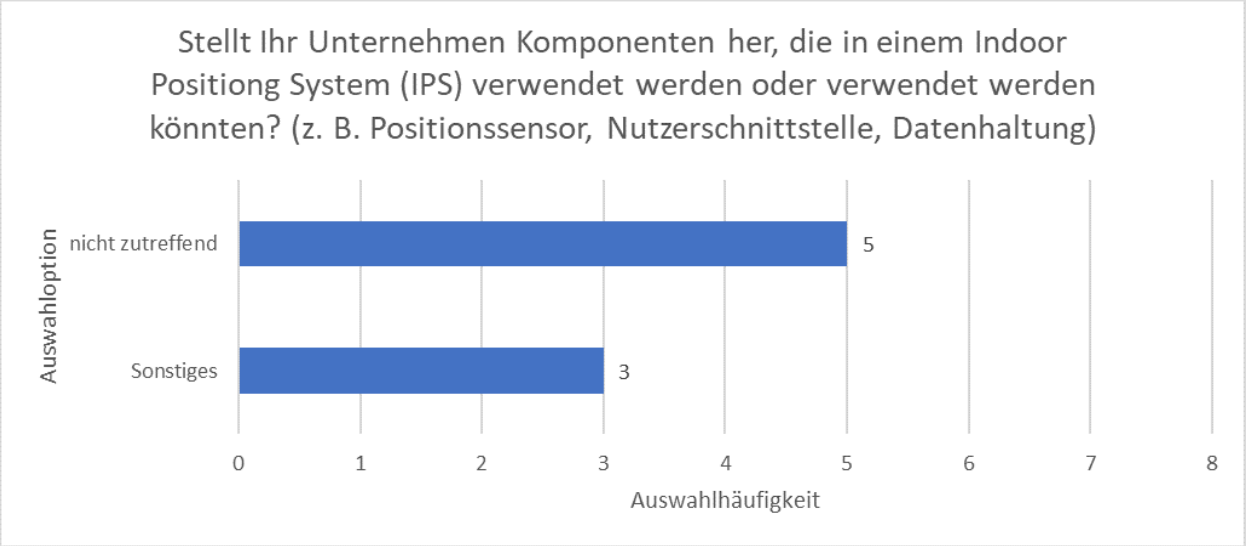

Abbildung 21 IPS-Bezug der Produkte des Unternehmens

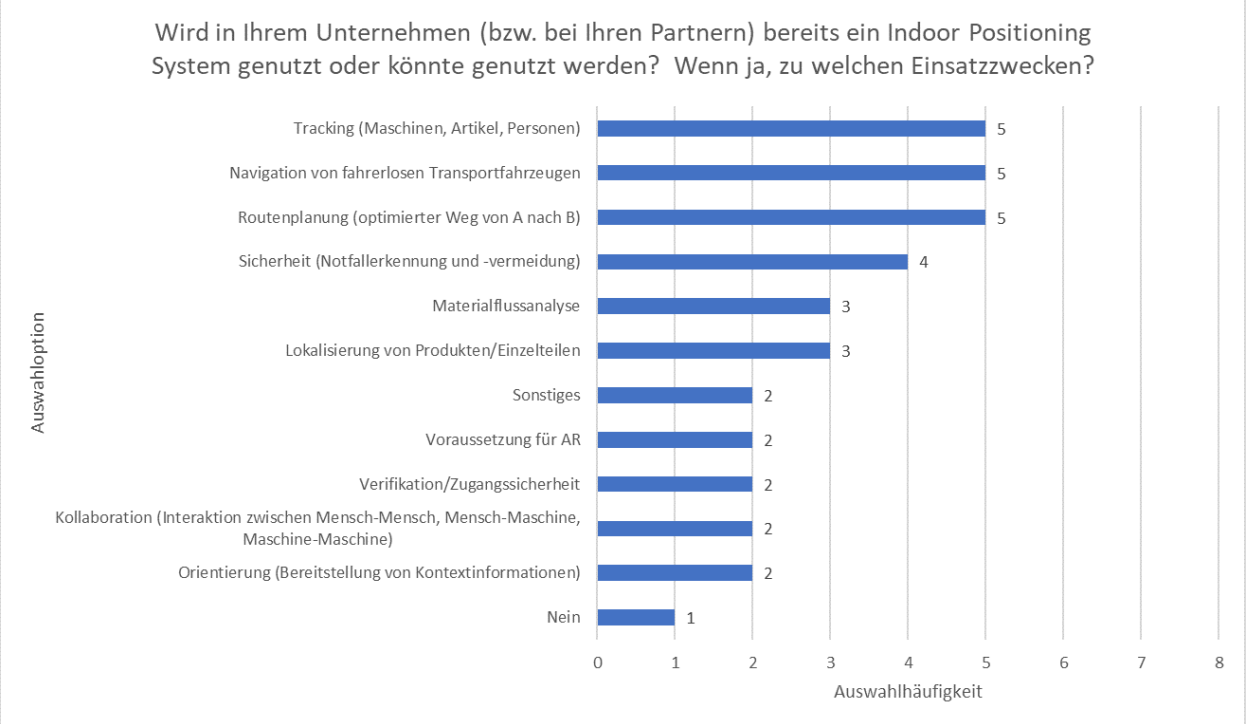

Abbildung 22 Genutzte IPS-Funktionalitäten

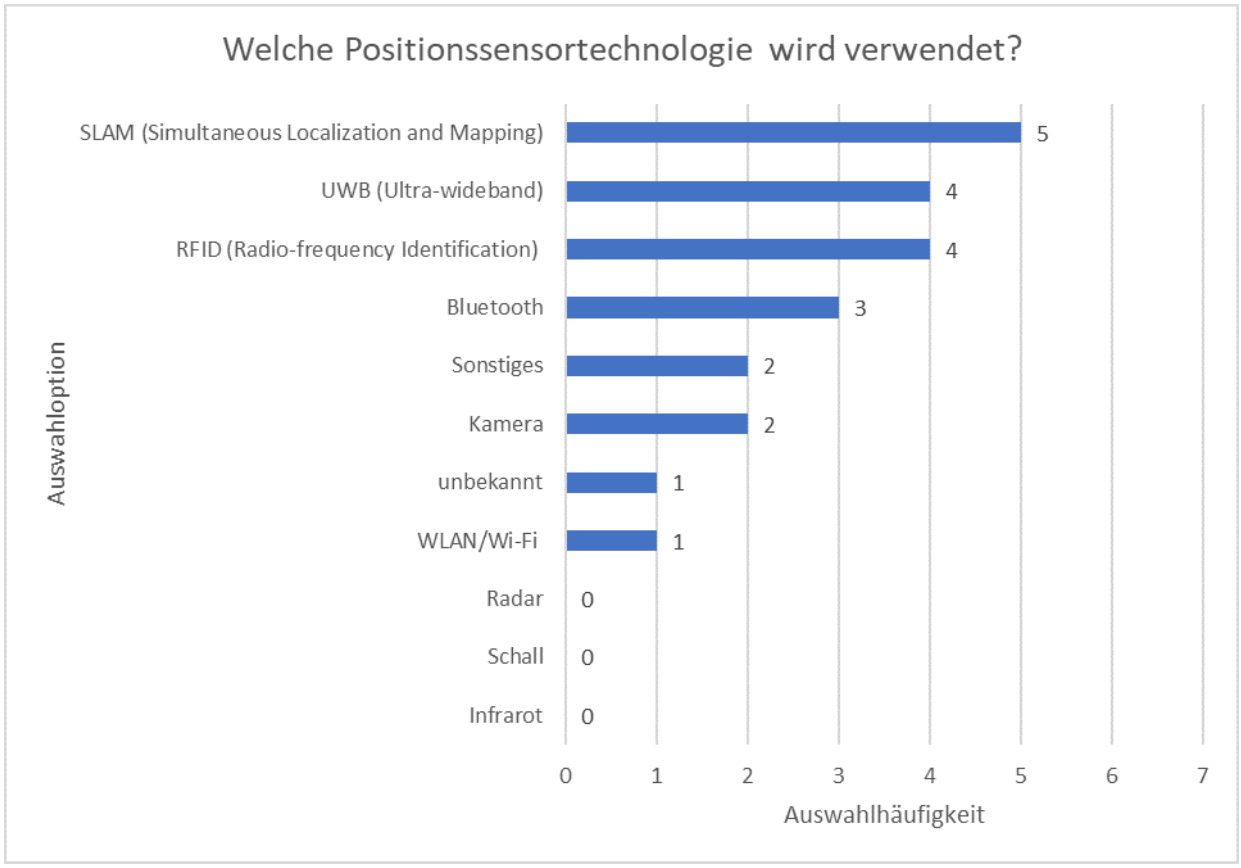

Abbildung 23 Verwendung Positionssensortechnologien

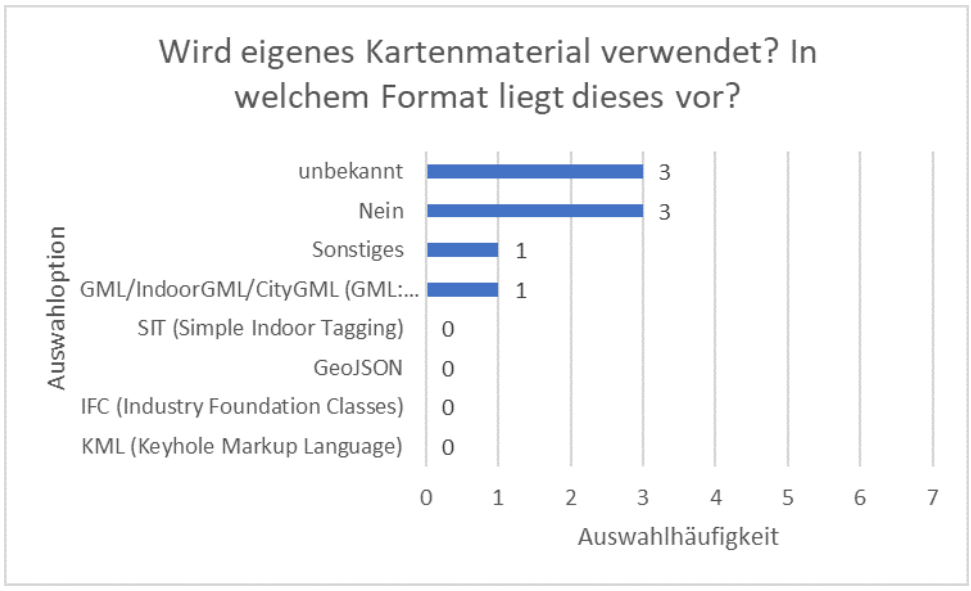

Abbildung 24 Verwendung Kartenmaterial

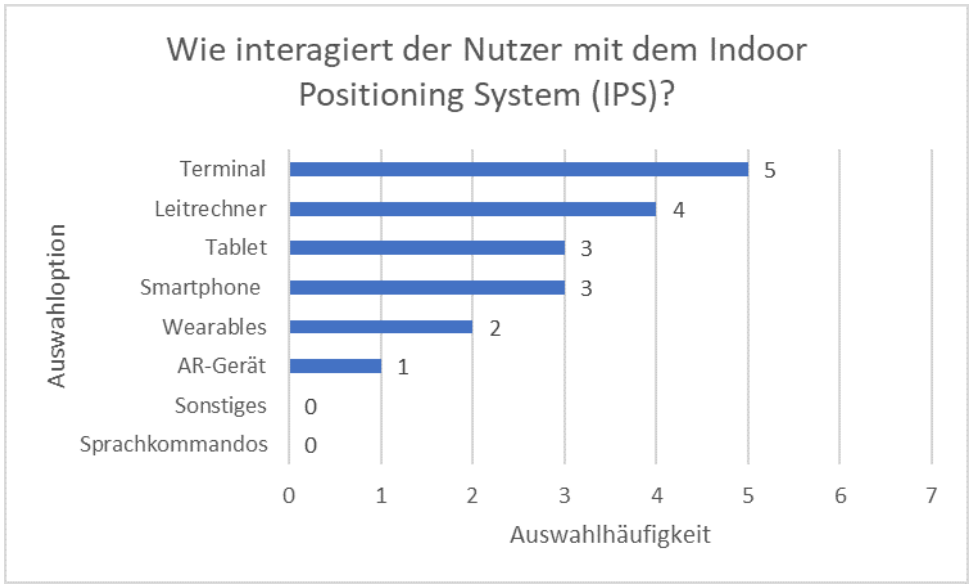

Abbildung 25 Nutzerinteraktion

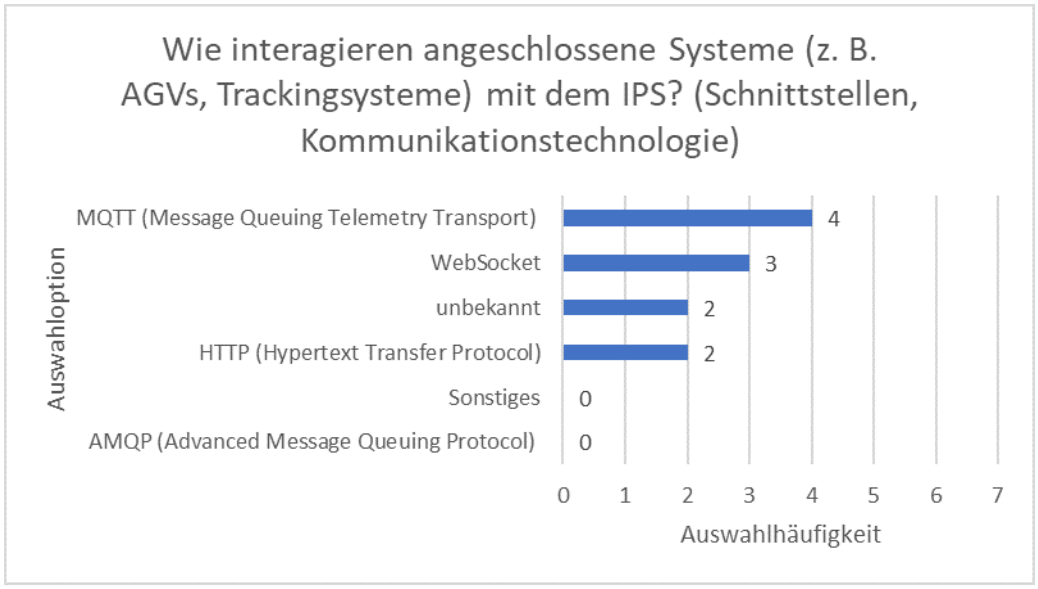

Abbildung 26 Interaktion angeschlossener Systeme

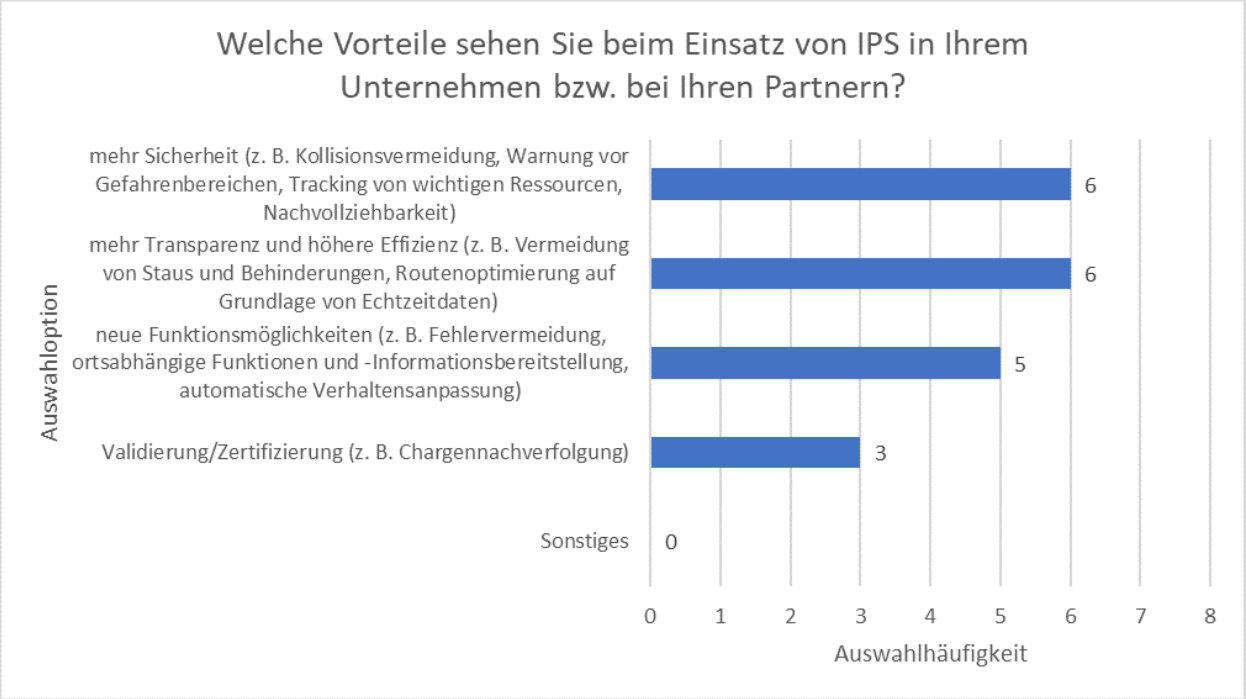

Abbildung 27 Einsatzvorteile IPS

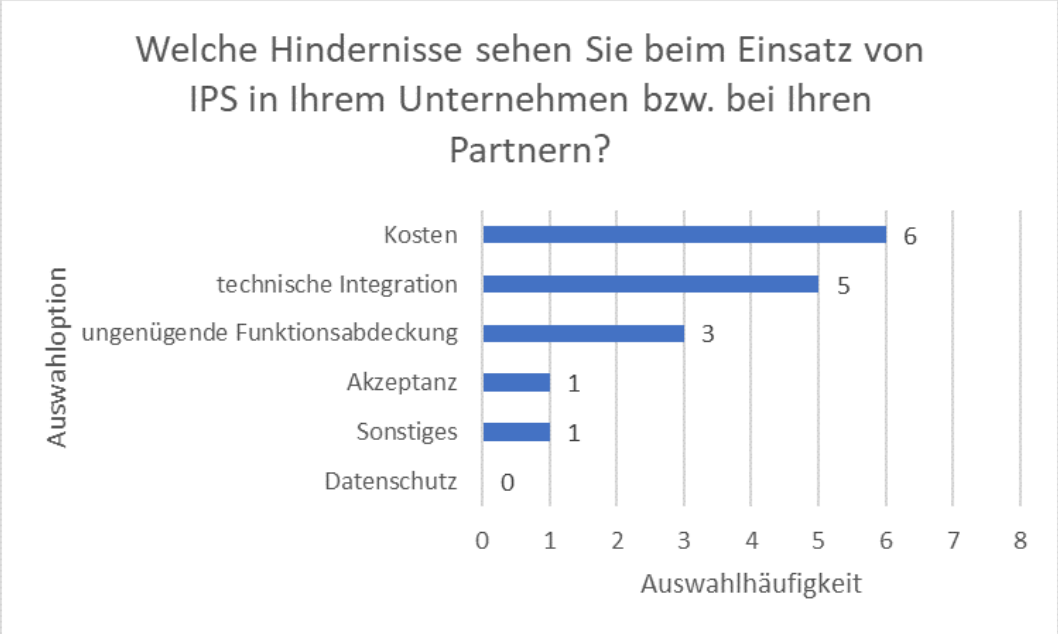

Abbildung 28 Einsatzhindernisse für IPS

Auf der Grundlage der Ergebnisse dieser Umfrage können wir im Rahmen der Bewertung des Projektergebnisses eine hohe Praxisrelevanz der vom IPos-FW bis zum Projektende unterstützten Technologien und Funktionalitäten feststellen. Insbesondere die Kommunikationstechnologie MQTT und die Positionssensortechnologie UWB besitzen eine hohe Praxisrelevanz. Die in der Fallstudie *Industrierobotik* untersuchten Möglichkeiten zur IPS-basierten Erhöhung der Sicherheit im Produktionsprozess wurden in mehreren Fragen mit einer hohen Häufigkeit genannt. Die in den Fallstudien *Orderpicker* und *Sensordatenfusion* untersuchten Möglichkeiten zur Erhöhung der Transparenz und Fehlervermeidung wurden in der Umfrage als wichtige Vorteile beim Einsatz von IPS genannt. Schließlich ermöglicht die in

der Fallstudie *VDA5050* untersuchte Kommunikation zwischen AGV und AGV-Steuerung nach einem etablierten und standardisierten Kommunikationsprotokoll die Navigation von fahrerlosen Transportfahrzeugen. Auch diese Funktionalität ist aus der Umfrage als ein wichtiger Einsatzzweck von IPS hervorgegangen.

Bewertungsschema

In diesem Abschnitt wird eine Übersicht über die Kriterien vorgestellt, nach denen das IPos-FW unter Auswertung der durchgeführten Fallstudien bewertet wurde.

- *allg. Funktionalität*
    - Dieses Kriterium betrifft die Nutzung der vom IPos-FW bereitgestellten typischen IPS-Funktionalitäten
- *spezifische Funktionalität*
    - Über dieses Kriterium werden IPS-Funktionalitäten erfasst, die im Rahmen der jeweiligen Fallstudie zur Realisierung der angestrebten Funktionalität über eine Erweiterung des IPos-FW hinzugefügt werden mussten
- *Integration*
    - Dieses Kriterium erfasst den Aufwand für die Anbindung der Erweiterungen an das IPos-FW
- *Wiederverwendung*
    - Über dieses Kriterium wird die Eignung der in der jeweiligen Fallstudie neu entstandenen Komponenten zur Wiederverwendung in anderen Anwendungen erfasst

Die Erarbeitung des Bewertungsschemas erfolgte im Rahmen des Arbeitspaketes 7.2.

Anwendung des Bewertungsschemas

Tabellarische Übersicht

In diesem Abschnitt erfolgt eine Bewertung der einzelnen Fallstudien nach dem im letzten Abschnitt vorgestellten Bewertungsschema.

Industrierobotik

- *allg. Funktionalität*

    - Positionsberechnung

    - Überwachungsaufträge

    - Koordinatentransformation

- *spezifische Funktionalität*

    - Keine (benötigte spezifische Funktionalität wurde von der (Bestands-)Software des Roboterarmes bereitgestellt)

- *Integration*

  - Aufwand gering; Bestandssoftware wurde um Unterstützung der klar definierten API des IPos-FW erweitert

- *Wiederverwendung*

  - Für diese Fallstudie mussten keine neuen Komponenten zur Erweiterung des IPos-FW entwickelt werden

Sensordatenfusion

- *allg. Funktionalität*

  - Positionsberechnung

  - Überwachungsaufträge

  - Sensordatenfusion

- *spezifische Funktionalität*

  - Graphische Darstellung von Positionen und Genauigkeiten

- *Integration*

  - Integrationsaufwand gering durch Verwendung des *devkit*

- *Wiederverwendung*

  - Die neu entstandene graphische Benutzeroberfläche eignet sich zur Wiederverwendung. Sie wurde auch in der Fallstudie *VDA5050* wiederverwendet.

Orderpicker

- *allg. Funktionalität*

  - Positionsberechnung

  - Überwachungsaufträge

- *spezifische Funktionalität*

  - Anwendungsspezifische Filterung von Positionsnachrichten

- *Integration*

  - Integrationsaufwand gering durch Verwendung des *devkit*

- *Wiederverwendung*

  - Für die Entwicklung der anwendungsspezifischen Filterung konnte auf der Grundlage einer im *devkit* vorhandenen Filterklasse erfolgen. Diese Filterklasse eignet sich durchaus zur Wiederverwendung.

VDA5050

- *allg. Funktionalität*
  - Überwachungsaufträge
  - Extraktion von Informationen aus Nachrichten etablierter unterstützter Protokolle
  - Unveränderte Weiterleitung von Nachrichten nicht unterstützter Protokolle
- *spezifische Funktionalität*
  - Graphische Darstellung von Positionen und Genauigkeiten
  - Erweiterung eines AGV-Simulatoren
- *Integration*
  - Integrationsaufwand gering durch Verwendung des *devkit*
- *Wiederverwendung*
  - Zum Einsatz kam eine anwendungsspezifisch erweiterte Version der bereits im Rahmen der Fallstudie *Sensordatenfusion* verwendeten graphischen Benutzeroberfläche. Zumindest diese Benutzeroberfläche eignet sich zur Wiederverwendung.

Tooz

- *allg. Funktionalität*
  - Überwachungsaufträge
  - Rohdatenweiterleitung
- *spezifische Funktionalität*
  - Anwendungsspezifische Filterung der empfangenen Rohdaten
  - Weiterleitung der Informationen an einen Service in einem anwendungsspezifischen Datenformat und mit einem anwendungsspezifischen Kommunikationsprotokoll
- *Integration*
  - Integrationsaufwand gering durch Verwendung des *devkit*
- *Wiederverwendung*
  - Die im Rahmen der Erweiterung entstandene Unterstützung für das Kommunikationsprotokoll *STOMP* kann auch in anderen Anwendungen wiederverwendet werden

Ausführliche Einschätzung

Auf die tabellarische Übersicht folgen in diesem Abschnitt einige ergänzende Bemerkungen. Die durchgeführten Fallstudien haben gezeigt, dass mit dem IPos-FW die Integration

verschiedenartiger Indoorpositionssysteme und Bestandssoftware unterschiedlicher Art zu einem Indoor-Positionssystem möglich ist. Der Entwickler wird bei der Entwicklung eines derartigen IPS durch die Möglichkeiten des *devkits* deutlich entlastet.

- Das Konzept zur zonenbasierten Definition von Überwachungsaufträgen hat sich in den Fallstudien *Industrierobotik* und *Orderpicker* als nützlich erwiesen. Dieses Konzept erlaubt in der Fallstudie *Industrierobotik* einem Roboterarm die Berücksichtigung von Raumeigenschaften wie der räumlichen Nähe von Agenten bei der Entscheidungsfindung. In der Fallstudie *Orderpicker* ermöglicht dieses Konzept im Rahmen der Überprüfung der Korrektheit des Kommissioniervorganges die Definition des genauen Standortes von Behältern.
- Das IPos-FW ist offen für die Integration von Sensoren unterschiedlicher Positionssensortechnologien. In der Fallstudie *Sensordatenfusion* wird ein Agent mit zwei Positionssensoren unterschiedlicher Positionssensortechnologien eingesetzt: UWB und NFC. Die vom IPos-FW bereitgestellte Sensordatenfusion ermöglicht grundsätzlich die Berücksichtigung aller Positionierungssensoren für einen Agenten zur Verfügung stehenden Positionssensoren bei der Positionsermittlung. In dieser Fallstudie erfolgte eine Auswahl der Positionssensorwerte nach der Genauigkeit der jeweiligen Positionssensortechnologie.

  Das IPos-FW ist offen für die Unterstützung von mehreren gleichartigen Positionssensortechnologien. So werden in den Fallstudien mehrere *Beacon-basierte* Positionssensortechnologien unterstützt: *UWB* (*Orderpicker, Sensordatenfusion, Industrierobotik*) und *Bluetooth* (*Tooz*).

- Das IPos-FW kann als technische Grundlage für die Entwicklung von Indoor-Positionssystemen (IPS) mit unterschiedlichster Geschäftslogik verwendet werden. In den Fallstudien wurden über das IPos-FW grundlegende Funktionalitäten eines IPS mit anwendungsspezifischen Funktionalitäten kombiniert.

  In der Fallstudie *Orderpicker* wurde unter Verwendung der Überwachungsaufträge des IPos-FW ein IPS mit einer anwendungsspezifischen Filterung von Positionsereignissen zur Korrektheitsprüfung des Kommissioniervorganges realisiert. In dieses auf der Grundlage des IPos-FW entstandene IPS wurde ebenfalls Unterstützung für einen Ausschnitt des Datenmodells des ERP-Systems *OFBiz* eingebaut. Dies zeigt auch die grundsätzliche Offenheit des IPos-FW für die Unterstützung von ERP-Systemen.

  In der Fallstudie *Tooz* werden Überwachungsaufträge des IPos-FW zur Ermittlung von Abstandsinformationen zu bestimmten Referenzpunkten für den ortsabhängigen Abruf von Umgebungsinformationen eingesetzt.

- Das IPos-FW ist offen für die Integration unterschiedlicher Aktortechnologien in ein IPS. Diese Eigenschaft des IPos-FW wurde in den Fallstudien für verschiedene Aktortechnologien gezeigt. In der *Industrierobotik*-Fallstudie wurde das IPos-FW verwendet, um das Weltmodell eines Roboterarms zu aktualisieren. Der Roboterarm benötigt diese Informationen, um einen sicheren Betrieb unter Berücksichtigung des aktuellen Standorts von Mitarbeitern im Produktionsprozess oder anderen Fahrrobotern durchzuführen.

In der Fallstudie *VDA5050* wurde das IPos-FW um die Unterstützung eines Ausschnittes des etablierten und standardisierten Protokolles *VDA5050* für die Kommunikation zwischen AGV und AGV-Steuerung erweitert. Dies ermöglicht insbesondere die Kommunikation von *VDA5050* kompatiblen AGVs mit *VDA5050*-inkompatiblen Leitstellen über das IPos-FW.

In der Fallstudie *Tooz* wurde das IPos-FW im Rahmen der Weiterleitung von Informationen an Datenbrillen des Herstellers *tooz technologies GmbH* eingesetzt. Die Informationen wurden an einen mit der Datenbrille verbundenen Dienst weitergeleitet, der anhand dieser Informationen die auf der Datenbrille anzuzeigenden Informationen ermittelt.

- Das im IPos-Projekt entstandene *devkit* hat sich im Rahmen der Fallstudienbearbeitung bei der Entwicklung der Erweiterungen des IPos-FW zu einem IPS als nützlich erwiesen. Die Klasse *IPosExtension* wurde in den Fallstudien *Orderpicker*, *Tooz*, *Sensordatenfusion* und *VDA5050* als Basisklasse für die Anbindung der Erweiterung an das IPos-FW verwendet. Die Verwendung der vom *devkit* bereitgestellten Transformationsfunktionen hat die Anbindung der Erweiterungen an das IPos-FW ebenfalls erleichtert.

Zusammenfassung und Abschluss

Zusammenfassung

Im Forschungsprojekt Indoor Positioning Software Framework for Intralogistics Applications (IPos) wurde die Entwicklung von Indoor-Positionssystemen (IPS) auf Basis eines Frameworks untersucht, insbesondere die damit verbundenen Möglichkeiten zur Erhöhung der technologischen Flexibilität von IPS. Entwickelt wurde ein Softwareframework, welches dem IPS-Entwickler klare Schnittstellen zur Erweiterung des Frameworks zu einem IPS zur Verfügung stellt. Dem Strukturprinzip der *Schichtenarchitektur* folgend sind alle Komponenten in Schichten mit klaren Verantwortlichkeiten organisiert, woraus sich ebenfalls gute Voraussetzungen für eine spätere Erweiterung der Kernfunktionalitäten des Frameworks ergeben. Der praktische Nutzen der vorhandenen Kernfunktionalitäten sowie der Aufwand einer IPS-Entwicklung auf Basis des Frameworks wurde in fünf Fallstudien untersucht. In diesen fünf Fallstudien wurden IPS zur technischen Unterstützung unterschiedlichster Szenarien aus der Domäne der intralogistischen Anwendungen entwickelt. Dabei wurde vom Framework die Verwendbarkeit für die Realisierung verschiedenartiger Funktionalitäten, einer technologischen Offenheit für unterschiedliche Positionssensor- sowie Aktortechnologien im Rahmen der IPS-Entwicklung verlangt. Die in diesem Bericht vorgestellten Ergebnisse zeigen, dass der Ansatz der Framework-basierten Entwicklung ein vielversprechender Ansatz für die Entwicklung von Indoor-Positionssystemen mit einer hohen technologischen Flexibilität ist, der für die Anwender einen Weg aus der technologischen Starrheit existierender Indoor-Positionssysteme sowie der durch das Problem des *vendor lock-ins* entstehenden wirtschaftlichen Zwangslage weist.

Mögliche weiterführende Arbeiten

In diesem Abschnitt werden einige Punkte genannt, die im Rahmen einer Fortführung der in diesem Forschungsprojekt begonnenen Arbeiten betrachtet werden könnten.

- Die Integration von IPos-FW und Erweiterungen erfolgt Nachrichten-orientiert über das Kommunikationsprotokoll MQTT. Obwohl sich dieser Ansatz bewährt hat, wäre es auch denkbar, den IPS-Entwicklern eine Erweiterung des IPos-FW in den unteren Schichten zu ermöglichen. Insbesondere wäre eine Implementierung eigener Filterbedingungen durch den IPS-Entwickler sinnvoll. Dazu müsste in der Komponente EventFilter (Funktionalitätsschicht) eine allgemeine Schnittstelle für die Auswertung von Filterbedingungen definiert werden und es dem IPS-Entwickler ermöglicht werden, eigene Implementierungen (Java) dieser Schnittstelle im IPos-FW zu registrieren.

- In Ergänzung zum *devkit* könnten Anwendungen für unterschiedliche Plattformen (Android, iOS) bereitgestellt werden, deren Verantwortlichkeit darin besteht, die von den Positionssensoren unterschiedlicher Positionssensortechnologien ausgesendeten Signale entgegennehmen zu können und an das IPos-FW weiterleiten zu können bzw. zur Weiterverarbeitung auf dem Smartphone bereitstellen zu können.

- Das *devkit* könnte für Anwendungsfälle erweitert werden in denen eine Erweiterung des IPos-FW zu entwickeln ist, welche Daten entgegennimmt und diese in bestimmte Überwachungsaufträge für das IPos-FW umsetzt. Für diese Fälle könnte das *devkit* um die Möglichkeit erweitert werden, aus einem gegebenen Datenmodell (welches die an die Erweiterung gesendeten Daten beschreibt) Code für den Empfang und die Weiterverarbeitung dieser Daten zu generieren. Durch diesen Generationsvorgang würde der IPS-Entwickler sehr schnell eine Erweiterung für das IPos-FW erhalten, die Daten empfangen kann, relevante Informationen extrahieren kann, IPos-Überwachungsaufträge erzeugen und auch an das IPos-FW versenden kann. Zur Beschreibung des Datenmodells würde sich z. B. das *Eclipse Modeling Frameworks* anbieten.

Aufgrund der Komplexität der behandelten Thematik konnte im Rahmen des Arbeitspaketes 8.1 lediglich eine inhaltliche Abstimmung für eine mögliche Veröffentlichung in einer Fachzeitschrift oder auf einer Konferenz erreicht werden. Diese Arbeiten sollen aber auch über die Projektlaufzeit hinaus weitergeführt werden. Eine Veröffentlichung des im Projekt erreichten wissenschaftlich-technischen Fortschritts in einem geeigneten Publikationsorgan wird ausdrücklich angestrebt. Im Hinblick auf diese Veröffentlichung ist ebenfalls eine Überarbeitung der Projektergebnisse unter [IPos_GitLab_protobuf] zur Verbesserung von deren Zugänglichkeit für eine Weiterentwicklung vorgesehen.

Notwendigkeit und Angemessenheit der geleisteten Arbeit

Die durchgeführten Arbeiten leisteten einen angemessenen Beitrag zum Forschungsvorhaben. Sie waren in Inhalt und Umfang ein notwendiger Teilschritt zur Erreichung des geplanten Forschungsziels.

Die durchgeführten Forschungsarbeiten sowie die dafür aufgewendeten Ressourcen waren notwendig und angemessen, um die Ziele der Arbeitspakete des Forschungsprojektes zu erreichen. Die Ergebnisse des Forschungsprojektes entsprachen den Erwartungen des Projektantrags.

Nutzen und Anwendungsmöglichkeiten

Das Forschungsprojekt *IPos Software Framework: Indoor Positioning Software Framework for Intralogistics Applications* hat gezeigt, dass der Framework-basierte Ansatz grundsätzlich für die Entwicklung von Indoor-Positionssystemen (IPS) mit für KMU sehr vorteilhaften technischen Eigenschaften geeignet ist. Derartige IPS zeichnen sich durch eine hohe technologische Flexibilität aus und ermöglichen damit insbesondere KMU die Überwindung der auch als *vendor lock-in* bezeichneten Abhängigkeit von IPS-Herstellern mit einem großen Produktangebot, die versuchen, ihre Kunden auf eigene Softwarekomponenten zur Erweiterung des eigenen IPS technisch einzuschränken. Ein nach dem Framework-basierten Ansatz entstandenes IPS-Framework kann die technische Grundlage für ein Software-Ökosystem, mit dessen Artefakten sich Indoor-Positionssysteme unterschiedlichster Funktionalitäten kostengünstig realisieren lassen. Dies ermöglicht den KMU eine schnelle Anpassung ihrer IPS in Reaktion auf veränderte Anforderungen oder auch eine aufwandsarme Modernisierung ihrer bestehenden IPS durch die Integration neuer Technologien. Die in diesem Forschungsprojekt entstandenen Ergebnisse können insbesondere als konzeptionelle Grundlage für die Entwicklung eines derartigen IPS-Frameworks verwendet werden.

Plan zum Ergebnistransfer in die Wirtschaft

Während der gesamten Projektlaufzeit wurde der intensive Kontakt mit den Mitgliedern des Projektbegleitenden Ausschusses gepflegt. Neben den zentralen Sitzungen des Projektbegleitenden Ausschusses lag der Schwerpunkt auf bilateralem Austausch über persönliche Gespräche und fernmündliche Kommunikation.

Tabelle 4 Durchgeführte Transfermaßnahmen

Maßnahme	**Beschreibung**	**Zeitraum**
Projektbegleitender Ausschuss	1. Sitzung des PA: Abstimmung PA Mitglieder, Vorstellung und Diskussion der Umfrageergebnisse, Diskussion zur Ausrichtung/Schwerpunktsetzung, Vorstellung erster Ergebnisse und Lösungsansätze	*01.03.2021*
Projektbegleitender Ausschuss	2. Sitzung des PA: Vorstellung des IPos Software Frameworks, Vorstellung Demonstratoren, Diskussion über mögliche Anwendungen	*29.11.2021*

Maßnahme	Beschreibung	Zeitraum
Internetpräsentation	https://tu-dresden.de/ing/informatik/smt/st/forschung/forschungsprojekte?project=60&embedding_id=47eddfa7c5a54ed5be49042aff35a31b	*Seit 06/2021*
Bereitstellung Implementierungen	Bereitstellung implementiertes IPos-Framework in einem online Repository: https://git-st.inf.tu-dresden.de/ipos-public	*Seit 06/2022*
Vorträge	Forschungsseminar der Professur für Technische Logistik Forschungsseminar der Professur für Softwaretechnologie	*01.03.2022* *01.03.2022*
Publikationen	Darstellung der Forschungsergebnisse in Fachzeitschriften mit Breitenwirkung in der Wirtschaft, z. B.: o Hebezeuge und Fördermittel o Fördern und Heben o Logistik Heute sowie ggf. auch in wissenschaftlichen Zeitschriften, z. B.: o Logistics Journal o Logistics Research o IEEE Transactions on Software Engineering Journal of Software Engineering Research and Development	*fortlaufend im Projekt*
Beratung von Unternehmen	In Kombination zu den Sitzungen des pbA wurden fortlaufend im Projekt zahlreiche bilaterale Gespräche geführt, bei denen Erkenntnisse kommuniziert und Ergebnisse analysiert wurden.	*fortlaufend im Projekt*
Analyse und Bewertung	Ein Fragebogen und Interview sind vorbereitet und wurden in Q1 2021 an den PA (und weitere) Unternehmen gerichtet. Analyse und Bewertung von IPos Software Framework durch Demonstratoren im Feld und Diskussion mit Unternehmen des PA.	*fortlaufend im Projekt*

Zusätzlich liefert Tabelle 4 eine Übersicht zum Stand unterschiedlichster Transfermaßnahmen während der Projektbearbeitung und nach Projektabschluss. Ein neuer Projektantrag zu einem Folgeprojekt, das insbesondere noch vorhandene Hürden bei der Umsetzung des entwickelten Lane-Konzepts bei kmU abbauen soll, ist zeitnah geplant.

Durchgeführte Transfermaßnahmen

Die Durchführung von Transfermaßnahmen wurde im Rahmen der Arbeitspakete 8.1 und 8.2 vorgenommen.

Verwendung der Zuwendung

Die Verwendung der Zuwendung entfiel ausschließlich auf Personalkosten. Eine Beschaffung von Ausrüstung und Technik war nicht vorgesehen und auch nicht nötig. Eine Aufschlüsselung der Personalkosten auf die beiden Forschungsstellen (Professur für Technische Logistik – TL bzw. Professur für Softwaretechnologie – SWT) findet sich in Tabelle 5.

Tabelle 5 Aufschlüsselung der Zuwendungen nach den Forschungsstellen

Zeitraum	**Verwendung der Zuwendung**	
	FST1: TL	FST2: SWT
2020	8 PM	6,49 PM
2021	12PM	15 PM
2022	7 PM	3,5 PM
Gesamt	**27 PM**	**24,99 PM**

Referenzen

Al-Ammar, Mai A.; Alhadhrami, Suheer; Al-Salman, Abdulmalik [et al.]: Comparative survey of indoor positioning technologies, techniques, and algorithms. 2014 International Conference on Cyberworlds. IEEE: 2014, pp. 245–252.

Brena, Ramon F.; García-Vázquez, Juan Pablo; Galván-Tejada, Carlos E. [et al.]: Evolution of indoor positioning technologies: A survey. In: Journal of Sensors vol. 2017 / 2017.

Eclipse Foundation (2022): Eclipse Modeling Framework (EMF). available at: https://www.eclipse.org/modeling/emf/.

IPos_GitLab: IPos Framework Repository. available at: https://git-st.inf.tu-dresden.de/ipos-public.

OFBiz: OFBiz Repo. available at: https://github.com/apache/ofbiz.

Rohde, Rohde; Zhu, Hailong: GenericSensor Protobuf Definition. available at: https://git-st.inf.tu-dresden.de/ipos-public/models/-/tree/main/interfaces.

VDA (2021): VDA5050. available at: https://github.com/VDA5050/VDA5050.