Verteilte Systeme

VS Praktikum SoSe 2025

Manh-An David Dao, Philipp Patt, Jannik Schön, Marc Siekmann

11. Mai 2025

Inhaltsverzeichnis

1 Einführung und Ziele **4**
- 1.1 Aufgabenstellung . . . 4
- 1.2 Qualitätsziele . . . 5
- 1.3 Stakeholder . . . 7

2 Randbedingungen **8**
- 2.1 Technische Randbedingungen . . . 8
- 2.2 Organisatorische Randbedingungen . . . 8

3 Kontextabgrenzung **9**
- 3.1 Fachlicher Kontext . . . 9
- 3.2 Technischer Kontext . . . 9
- 3.3 Externe Schnittstellen . . . 9

4 Lösungsstrategie **11**

5 Bausteinsicht **12**

6 Laufzeitsicht **13**
- 6.1 Szenario I . . . 13
- 6.2 Szenario II . . . 14
- 6.3 Szenario III . . . 15

7 Verteilungssicht **16**

8 Konzepte **17**
- 8.1 Offenheit . . . 17
- 8.2 Verteilungstranzparenzen . . . 17
- 8.3 Kohärenz . . . 17
- 8.4 Sicherheit (Safety) . . . 17
- 8.5 Bedienoberfläche . . . 17
- 8.6 Ablaufsteuerung . . . 17
- 8.7 Ausnahme- und Fehlerbehandlung . . . 18
- 8.8 Kommunikation . . . 18
- 8.9 Konfiguration . . . 18
- 8.10 Logging, Protokollierung . . . 18
- 8.11 Plausibilisierung und Validierung . . . 18
- 8.12 Sessionbehandlung . . . 18
- 8.13 Skalierung . . . 18
- 8.14 Verteilung . . . 18

9 Entwurfsentscheidungen **19**

10 Qualitätsszenarien **20**

11 Risiken **21**

1 Einführung und Ziele

Diese Dokumentation beschreibt Entwurf und Kontext einer Middleware, die im Rahmen des Praktikums „Verteilte Systeme SoSe 2025“ entwickelt wird. Ziel des Praktikums ist der Entwurf und die Implementierung eines verteilten Steuerungssystems für autonome Roboterarme. Die Middleware bildet dabei eine zentrale Komponente, die die Komplexität der Verteilung für die Anwendungsebene abstrahieren soll.

1.1 Aufgabenstellung

Die Hauptaufgabe der zu entwickelnden Middleware besteht darin, die Kommunikation und Koordination zwischen den verteilten Komponenten des Steuerungssystems zu ermöglichen. Sie soll eine Abstraktionsschicht zwischen der eigentlichen Anwendungslogik (z.B. Robotersteuerung) und den darunterliegenden Betriebssystem- und Netzwerkdiensten bilden. Gemäß den Prinzipien verteilter Systeme nach Tanenbaum & van Steen, die auch im zugrundeliegenden Skript referenziert werden, verfolgt die Middleware das Ziel, die Verteilung weitestgehend zu verbergen (Transparenz). Zu den Kernfunktionen, die eine Middleware typischerweise bereitstellt und die hier adressiert werden sollen, gehören:

- Bereitstellung eines breiten Angebots für die Kommunikation mittels verschiedener Protokolle. Dies kann synchrone oder asynchrone Kommunikation umfassen, wie z.B. Remote Procedure Calls (RPC) oder Nachrichtenübertragung (Message-Oriented Middleware, MOM).
- Unterstützung zur Konvertierung von Datenformaten zwischen Systemen.
- Bereitstellung von Protokollen zur Namensauflösung (Service Discovery), um Ressourcen einfach zu identifizieren und zu referenzieren.
- Mechanismen der Skalierbarkeit, wie z.B. Replikation

Die Middleware dient als Vermittler, der die Interoperabilität über verschiedene Systeme, Betriebssysteme und Hardwaretypen hinweg verbessern soll.

1.2 Qualitätsziele

Tabelle 1.1: Qualitätsziele der Software Engineering

Ziel	Beschreibung	Metrik
Zuverlässigkeit	Fehler dürfen den Betrieb nicht gefährden. Fehlererkennung und -toleranz müssen integriert sein.	Das System ist über dem gesamten Abnahmezeitraum stabil (ca. 1,5 h)
Skalierbarkeit	Zusätzliche Roboter oder Komponenten sollen ohne Änderungen an der bestehenden Architektur integrierbar sein. Die Middleware muss Mechanismen (z.B. Replikation, Caching) bieten, um mit zunehmender Anzahl von Teilnehmern oder der Datenmenge zurechtzukommen.	Es können beliebig viele Roboter hinzugefügt und entfernt werden (0 - 253)
Sicherheit (Safety)	Es bewegt sich immer genau ein Roboterarm. Sollte das System nicht wie gewünscht reagieren, wird ein sicherer Zustand erreicht	Reaktionszeit max. 250 ms, bis Roboterarm stoppt
Wartbarkeit	Der Code muss übersichtlich sein, gut dokumentiert sein und wenig Komplexität enthalten.	Zyklomatische Komplexität ≤ 10 und LOC ≤ 30 pro Methode/Funktion exklusive Kommentar
Ressourcenteilung	Mehrere Roboterarme können über einem Netzwerk von einem ITS-Board gesteuert werden	Alle hinzugefügten Roboterarme werden zuverlässig angesteuert

Ziel	Beschreibung	Metrik
Offenheit	Zugänglichkeit Interoperabilität Portabilität	Einzelne Softwarekomponenten können, ausgetauscht oder portiert werden, um zb Standards bzgl. Kommunikation und Datenstruktur zu tauschen oder um die Plattform zu wechseln. Die Funktionalität bleibt gleich
Zugriffstransparenz	Die Umsetzung der Roboterarmsteuerung ist für den Benutzer nicht erkennbar und möglichst einfach gehalten	GUI zeigt nur Zustände des ausgewähltem Roboterarms an (z.B. Verfügbarkeit) und der Roboterarm ist über das ITS-Board so einfach wie möglich steuerbar
Lokalitäts-Transparenz	Die Netzwerk- und Softwarestruktur ist nach außen unsichtbar	Das Interface nach außen ist eineiheitlich und verschleiert die Implementierung

1.3 Stakeholder

Tabelle 1.2: Interessen der Stakeholder

Stakeholder	**Interesse**
Nutzer	• Indirekte Stakeholder, deren Erfahrung stark von der Leistung und Zuverlässigkeit des Gesamtsystems, einschließlich der Middleware, abhängt.
Betreiber	• Portabilität: Das System kann auf verschiedenen Plattformen betrieben werden. • Zuverlässigkeit: Die Middleware kann über den gesamten benötigten Zeitraum ohne Ausfälle genutzt werden
Entwicklerteam Middleware	• Wartbarkeit • Portabilität: Das System kann auf verschiedenen Plattformen betrieben werden (z.B Testen) • Austauschbarkeit: Softwaremodule können ohne großen Aufwand ersetzt werden
Entwicklerteam Applikation	• vollständige Funktionalität • Benutzerfreundlichkeit (der Middleware-Schnittstelle) • Zuverlässigkeit und Reaktionszeit der von der Middleware bereitgestellten Kommunikationsdienste.
Professor	• Zugang zu allen Arbeitsmitteln zwecks Bewertung und Kontrolle • Das Endprodukt besitzt alle geforderten Funktionalitäten

2 Randbedingungen

Dieses Kapitel beschreibt die Rahmenbedingungen, unter denen die Middleware entworfen und implementiert werden muss.

2.1 Technische Randbedingungen

- **Betriebsumgebung:** Die Middleware muss auf einer heterogenen Umgebung aus verschiedenen Hardware-Plattformen und Betriebssystemen laufen können. Im spezifischen Projektkontext umfasst dies mindestens ein ITS-Board (STM32F4) und mehrere Raspberry Pis.
- **Netzwerk:**
  - Die Kommunikation findet innerhalb eines begrenzten Netzwerks statt, im Projektkontext ein /24 Netzwerk. Die Middleware baut auf den grundlegenden Kommunikationsdiensten des Betriebssystems und Netzwerks auf.
- **Hardware:** Die Middleware muss auf der Hardware laufen, die für das Steuerungssystem verwendet wird: ITS-Board und Raspberry Pis. Dies impliziert möglicherweise Beschränkungen hinsichtlich Rechenleistung, Speicher und verfügbaren Bibliotheken im Vergleich zu leistungsstarken Servern.
- **Anbindung:** Die Middleware muss in der Lage sein, sich mit der Anwendungsschicht darüber und den System-/Netzwerkschichten darunter zu integrieren.

2.2 Organisatorische Randbedingungen

- **Zeit:** Entwicklungszeitraum beträgt 12 Wochen.
- **Vorwissen:** Einige Konzepte und Herangehensweisen werden erst im Laufe der 12 Wochen gelernt.
- **Budget:** Es steht kein Budget zur Verfügung.

3 Kontextabgrenzung

3.1 Fachlicher Kontext

Die Middleware ist fachlich im Bereich der verteilten Steuerungssysteme angesiedelt. Ihre Rolle besteht darin, die komplexen Aspekte der Verteilung für die darüberliegende Anwendungslogik zu kapseln und zu abstrahieren. Sie ist nicht die Steuerungslogik selbst, sondern das Rückgrat für die Kommunikation und Koordination dieser Logik, die auf mehreren verteilten Komponenten (ITS-Board, Raspberry Pis) ausgeführt wird. Sie ermöglicht es, dass beliebig viele (1 - 254) Roboterarme von einem zentralen ITS-Board aus gesteuert werden können, indem sie die Nachrichtenübermittlung und den Austausch von Steuerungsbefehlen handhabt. Sie muss sicherstellen, dass die Kommunikation auch unter den Bedingungen des Projekts (z.B. Fehlverhalten von Komponenten) die Sicherheit der Anwesenden nicht gefährdet, was eine Anforderung an die Zuverlässigkeit und Fehlertoleranz der Middleware impliziert.

3.2 Technischer Kontext

Technisch gesehen positioniert sich die Middleware als eine Schicht zwischen der Anwendungsebene und den Betriebssystem-/Kommunikationsdiensten. Sie ist logisch über mehrere Maschinen verteilt, die über ein Netzwerk (hier: /24 Netzwerk) verbunden sind. Sie nutzt die unterliegenden Netzwerkprotokolle (z.B. TCP/UDP) und Betriebssystemfunktionen, bietet aber eine höhere Abstraktionsebene für die Anwendungen. Im spezifischen Projektkontext läuft sie auf den genannten Hardware-Komponenten (ITS-Board, Raspberry Pis). Die Middleware muss die Heterogenität dieser zugrundeliegenden Systeme (Hardware, Betriebssysteme) verbergen. Sie stellt Middleware-Kommunikationsprotokolle bereit.

3.3 Externe Schnittstellen

Die Middleware interagiert hauptsächlich über zwei externe Schnittstellenbereiche:

- Schnittstelle zur Anwendungsebene: Dies ist die primäre Schnittstelle der Middleware. Sie bietet eine abstrahierte, einheitliche API (Application Programming Interface) für die Anwendungsentwickler. Über diese Schnittstelle kann die Anwendungslogik (z.B. auf dem ITS-Board oder den Raspberry Pis) verteilte Dienste nutzen, Nachrichten senden/-empfangen oder auf benannte Ressourcen zugreifen, ohne sich um die Details der Netzwerkimplementierung, des Marshallings oder der physikalischen Lokalität kümmern zu müssen. Die Schnittstelle sollte idealerweise so gestaltet sein, dass sie sich nicht wesentlich von lokalen Funktionsaufrufen oder Datenzugriffen unterscheidet, um die Transparenz zu maximieren. Sie kann die Unterstützung mehrerer Programmiersprachen umfassen.
- Schnittstelle zu den System-/Netzwerkschichten: Die Middleware nutzt die Dienste des lokalen Betriebssystems und des Netzwerkstacks. Sie interagiert mit diesen Schichten, um

Nachrichten über das Netzwerk zu senden und zu empfangen, auf Ressourcen zuzugreifen und systemnahe Funktionen auszuführen. Diese Schnittstelle ist intern und verborgen für die Anwendungsebene.

Diese Struktur entspricht dem in den Quellen diskutierten Modell einer Middleware-Schicht, die über den Betriebssystemen liegt und Anwendungen eine einheitliche Schnittstelle bietet.

4 Lösungsstrategie

5 Bausteinsicht

6 Laufzeitsicht

6.1 Szenario I

Anschalten der Roboter-Software

6.2 Szenario II

Steuerbefehl über GUI-Software an einen ausgewählten, verfügbaren Roboter

6.3 Szenario III

Fehlerfall: Verbindung zu Roboter fällt aus

7 Verteilungssicht

8 Konzepte

8.1 Offenheit

8.2 Verteilungstranzparenzen

8.3 Kohärenz

Kohärenz wird dadurch sichergestellt, dass die Steuerung alle Roboter ausschliesslich über das ITS-Board zulässig ist.

8.4 Sicherheit (Safety)

Watchdogs auf dem Raspberry Pi des Roboterarms überwacht die eigene Verfügbarkeit. Wenn sich der Roboterarm durch einen vorherigen Aufruf vom ITS-Board bewegt, muss eine Verbindung zum ITS-Board bestehen. Bricht diese ab, oder das Netzwerk ist zu stark ausgelastet, muss der Roboterarm sofort anhalten. Wenn das ITS-Board einen Abbruch der Verbindung feststellt ist der Roboterarm als "nicht verfügbarërkennbar. Security ist keine Anforderung an das System.

8.5 Bedienoberfläche

Die GUI wird textuell auf dem Display des ITS-Boards dargestellt. Es wird auf Einfachheit und Eindeutigkeit geachtet.

8.6 Ablaufsteuerung

Ablaufsteuerung von IT-Systemen bezieht sich sowohl auf die an der (grafischen) Oberfläche sichtbaren Abläufe als auch auf die Steuerung der Hintergrundaktivitäten. Zur Ablaufsteuerung gehört daher unter anderem die Steuerung der Benutzungsoberfläche, die Workflow- oder Geschäftsprozessteuerung sowie Steuerung von Batchabläufen.

8.7 Ausnahme- und Fehlerbehandlung

8.8 Kommunikation

Für die Kommunikation wird das TCP-Protokoll genutzt. Die Steuerung der Roboterarme wird RPC durchgeführt.

8.9 Konfiguration

Die Raspberry Pi's und das ITS-Board bekommen eine feste IP-Adresse in einer bestehenden Netzwerkumgebung. Es existiert kein DHCP. Jeder Roboterarm hat eine eigene Hardgecodete Konfiguration, die dem ITS-Board mitgeteilt wird.

8.10 Logging, Protokollierung

Das Logging findet auf dem Raspberry Pi statt.

8.11 Plausibilisierung und Validierung

Wo und wie plausibilisieren und validieren Sie (Eingabe-)daten, etwa Benutzereingaben?

8.12 Sessionbehandlung

Sofern ein Roboterarm sich beim ITS-Board erfolgreich registriert hat, kann dieser, wenn erreichbar, angesteuert werden. Dafür wird eine TCP-Verbindung aufgebaut und danach mit RPC kommuniziert. Ein Watchdog auf beiden Seiten überwacht die Verbindung.

8.13 Skalierung

Dies Skalierung ist durch das /24 Netz begrenzt. Ansonsten können sich die Roboterarme beliebig mit ihren Kenndaten bei dem ITS-Board anmelden.

8.14 Verteilung

Das ITS-Board ruft per RPC die API der Middleware auf, diese wird dann in eine Bewegung des Roboterarms übersetzt. Die Nachrichten werden asynchron ausgetauscht. Die Anmeldung des Roboters erfolgt ebenfalls durch einen RPC aufruf. am ITS-Board.

9 Entwurfsentscheidungen

Entscheidung	Alternativen	Begründung	Woche

Tabelle 9.1: Zentrale Entwurfsentscheidungen

10 Qualitätsszenarien

11 Risiken