Verteilte Systeme

VS Praktikum SoSe 2025

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

29. April 2025

Inhaltsverzeichnis

1 Einführung und Ziele

1.1 Aufgabenstellung

Es soll ein verteiltes Steuerungssystem gemäß den Prinzipien verteilter Systeme nach Tanenbaum & van Steen entworfen und implementiert werden. Über ein ITS-Board (bspw. STM32F4) soll ein autonomer Roboter innerhalb eines kontrollierten Areals (z.B. BT7 R7.65 – als realer Testbereich) sicher und effizient gesteuert werden. Die verteilte Architektur soll dabei explizit gewährleisten, dass durch Fehlverhalten der Software oder Architektur keine Gefahr für anwesende entstehen kann.

1.2 Qualitätsziele

Das verteilte Steuerungssystem soll eine Reihe von nicht-funktionalen Anforderungen erfüllen, um einen sicheren, wartbaren und erweiterbaren Betrieb im Einsatzumfeld zu gewährleisten.

1.2.1 Ziele für Software Engineering

Ziel	**Beschreibung**	**Metrik**
Funktionalität		Alle Abnahmetests werden erfolgreich bestanden
Zuverlässigkeit		Das System ist über dem gesamten Abnahmezeitraum stabil (ca. 1,5 h)
Skalierbarkeit		Es können beliebig viele Roboter hinzugefügt und entfernt werden (0 - N)
Leistung		Reaktionszeit max. 500 ms
Sicherheit (Safety)		Reaktionszeit max. 500 ms
Wartbarkeit		
Portabilität		nicht relevant
Benutzerfreundlichkeit		Keine Einweisung erforderlich
Anpassbarkeit		
Kompatibilität		

Tabelle 1.1: Qualitätsziele der Software Engineering

1.2.2 Ziele der Verteilte Systeme

Ziel	**Beschreibung**	**Metrik**
Ressourcenteilung	...	
Offenheit	...	
Skalierbarkeit	...	siehe Tabelle 1.3
Verteilung Transparenz	...	siehe Tabelle 1.4

Tabelle 1.2: Qualitätsziele der Verteilten Systeme

Skalierbarkeit

Ziel	**Metrik**	**Metrik**
Vertikale Skalierung	...	
Horizontale Skalierung	...	
Räumliche Skalierbarkeit		1
Funktionale Skalierbarkeit	...	
Administrative-Skalierbarkeit		1

Tabelle 1.3: Skalierbarkeit von verteilten Systemen

Verteilungs-Transparenzen

Ziel	**Beschreibung**	**Metrik**
Zugriffstransparenz	...	
Lokalitäts-Transparenz	...	
Migrationstransparenz	...	
Replikationstransparenz	...	
Fehlertransparenz	...	
Ortstransparenz	..	
Skalierbarkeits-Transparenz	...	

Tabelle 1.4: Verteilungs-Transparenzen

1.3 Stakeholder

Die folgenden Gruppen sind direkt oder indirekt vom System betroffen:

- **Endnutzer:** Personen, die mit dem Roboter interagieren (z.B. Lehrpersonal, Kinder im Testbereich)
- **Entwicklerteam:** Zuständig für Entwurf, Umsetzung, Tests und Wartung des Systems
- **Betreiber:** Verantwortlich für die Überwachung und den sicheren Betrieb des Systems im realen Umfeld
- **Professor:** Person, die für die Bewertung des Systems verantwortlich ist und vorgabe der Rahmenbedingungen.

Use Case Diagramm

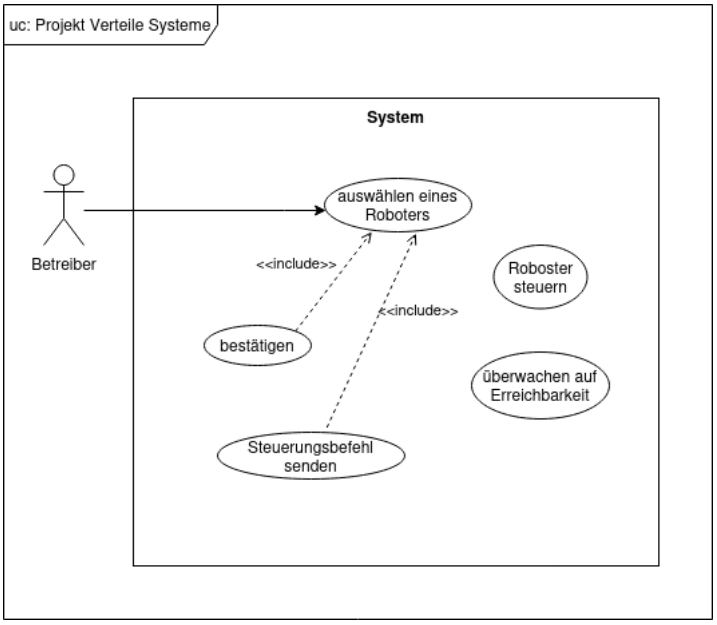

Abbildung 1.1: Funktionales Use Case Diagramm der Aufgabenstellung

2 Randbedingungen

2.1 Technische Randbedingungen

Das verteilte Steuerungssystem unterliegt mehreren festgelegten technischen Rahmenbedingungen, die den Entwicklungs- und Implementierungsspielraum einschränken. Diese Bedingungen sind im Folgenden aufgeführt:

- **Hardwareplattform:** Das System basiert auf ein Raspberry Pi der den jeweiligen Roboterarm steuert. Eine Software auf dem Raspberry Pi stellt eine API zur Verfügung, sodass die Roboter sich ohne Einschränkungen bewegen können. Ein ITS-BRD dient als Steuerung/Administration.
- **Programmiersprachen:**
  - **Java:** Die Roboterarme werden mit Javaprogramm angesteuert.
- **Netzwerkumgebung:** Es steht ein Adressenraum von 255 Adressen zur verfügung. Daraus folgt, dass nicht mehr als 254 Roboterarme eingesetzt werden können. Das ITS-Board ist ebenfalls ein Teilnehmer des Netzes.

2.2 Organisatorische Randbedingungen

- **Umgebung:** Der Abnahme bereich befindet sich im Raum BT7 R7.65. Die Steuerung und Navigation des Roboters müssen innerhalb der räumlich definierten Grenzen erfolgen. Die dort vorhandene LAN-Infrastruktur kann zur Netzwerkkommunikation verwendet werden.
- **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

Ziel dieses Kapitels ist es, das zu entwickelnde System innerhalb seines fachlichen und technischen Umfelds klar einzugrenzen. Dazu wird das System in Bezug auf seine Aufgaben (fachlicher Kontext), seine Einbettung in die bestehende technische Infrastruktur (technischer Kontext) sowie die definierten externen Schnittstellen beschrieben.

3.1 Fachlicher Kontext

Das verteilte System soll es ermöglichen beliebig viele (1 - 254) Roboterarme in einem Raum zu steuern. Die Kommunikation zum Nutzer und die Steuerung wird durch ein ITS-Board realisiert.

3.2 Technischer Kontext

Die Realisierung wird in einem /24 Netzwerk durchgeführt. In diesem wird das ITS-Board liegen. Ebenfalls sind in den Netzwerk die Roboterarme mit den vorgeschalteten Raspberry Pi, der jeweils einen Roboterarm ansteuert. Weitere Hilfmittel wie eine ICC-Cloud können ebenfalls genutzt werden.

3.3 Externe Schnittstellen

- System: Der Benutzer kann über ein Bildschirm einzelne erreichbare Roboterarme erkennen. Die Auswahl und Steuern der einzelnen Roboterarme wird durch Buttons realisiert.
- ITS-Board: Das ITS-Board kommuniziert über einen TCP-Server mit einer eigenen IP-Adresse mit dem System.
- Roboterarm: Der Roboterarm wird mit einer IP-Adresse angesprochen. Dafür ist der vorgeschaltete Raspberry Pi mit einem eigenem TCP-Server zuständig. Dieser stellt ebenfalls eine API für die Steuerung zur Verfügung.
- ICC Cloud: Die ICC Cloud kommuniziert über eine IP-Adresse.

4 Lösungsstrategie

Grundlage ist ein angepasstes Drei-Schichten-Modell, das auf die Anforderungen (KAPITEL) zugeschnitten ist.

- Präsentationsschicht: Diese besteht aus dem ITS-Board, das über ein integriertes Display eine grafische Benutzeroberfläche (GUI) zur Verfügung stellt. Die Interaktion erfolgt über Buttons, die Steuerbefehle auslösen. Das ITS-Board kommuniziert ausschließlich mit der Middleware und hat keine direkte Verbindung zu den Robotern.
- Logik- bzw. Vermittlungsschicht: Die Middleware ist das zentrale Element des Systems. Sie verwaltet den Systemzustand, koordiniert die Steuerbefehle und leitet Anfragen gezielt an die jeweiligen Roboter weiter. Zusätzlich fungiert sie als Watchdog: Sie überwacht alle aktiven Verbindungen (ITS-Board und Roboter) und sorgt im Fehlerfall (z.B. Verbindungsabbruch) für einen sicheren Stopp der Roboteraktivität. Die Middleware führt eine zentrale Registrierungstabelle, in der sich alle Teilnehmer (Roboter und ITS-Board) mit einem Typ-Tag (z.B. "robot", "controller") und einem eindeutigen Bezeichner (Name oder ID) anmelden müssen.
- Geräteschicht: Die Geräteschicht umfasst die Roboter selbst sowie deren lokale Steuereinheit und Sensorik. Jeder Roboter betreibt eine eigene API zur Steuerung und stellt Statusinformationen bereit. Auch auf dieser Ebene ist ein Watchdog implementiert, der bei Verbindungsverlust zur Middleware sofortige Sicherheitsmaßnahmen (z.B. Not-Stopp) auslöst.

Die Kommunikation zwischen den Schichten erfolgt über TCP/IP. Die Wahl von TCP garantiert eine zuverlässige, geordnete und fehlerfreie Datenübertragung. Um jedoch Verbindungsabbrüche rechtzeitig zu erkennen, wird zusätzlich ein Heartbeat-Mechanismus (Watchdog) eingesetzt, der regelmäßig geprüft wird (siehe Metriken, Kapitel). Alle Teilnehmer (ITS-Board und Roboter) müssen sich nach dem Systemstart aktiv bei der Middleware registrieren. Die Middleware verwaltet eine zentrale Teilnehmerliste in Form einer Tabelle mit Tupeln der Form `(Typ, Name, IP-Adresse)`. Diese wird zur Adressierung und gezielten Kommunikation verwendet.

In der GUI werden nur die erreichbaren Roboterarme aufgeführt. So kann sich der Benutzer einen der Roboterarme aussuchen und diesen anschließend per Buttons steuern. Dabei werden evtl. vorherige bewegte Roboterarme gestoppt. Sollte ein Roboterarm ausfallen bzw. vom Netz getrennt werden, ist dieser Roboterarm nicht mehr auf der GUI aufgeführt. Alle Bewegungen dieses Roboters werden sofort beendet. Ebenfalls wird das ITS-Board mit einem Not-Aus Button ausgestattet, sodass der ausgewählte Roboter sofort jede Bewegung abbricht.

Überprüfung der Lösungsstrategie

Ziele für Software Engineering

- **3-Schichten-Modell**: Klare Trennung von Darstellung, Logik und physikalischer Ausführung. Separation of Concern (SE-Ziele Funktionalität, Skalierbarkeit, Wartbarkeit, Anpassbarkeit, Kompatibilität)
- **Watchdog**: Middleware und Roboter überwachen gegenseitig den Verbindungsstatus, um bei Abbruch in einen sicheren Zustand zu wechseln. (SE-Ziel Safety)
- **Kommunikationsprotokoll:** TCP/IP wurde gewählt, um eine zuverlässige, verbindungsorientierte Kommunikation zu gewährleisten.
- **Adressierung:** Die Teilnehmer befinden sich im selben /24-Netzwerk und kommunizieren über IPv4. Jeder Roboter benötigt eine eindeutige ID oder Namen.

Ziele für Verteilte Systeme

- **Ressourcenteilung:** Alle Roboter können über eine einheitliche GUI des ITS-Boards gemeinsam verwendet werden.
- **Offenheit:** Neue Roboter können sich dynamisch an der Middleware registrieren. Das System bleibt erweiterbar ohne strukturelle Änderungen.
- **Skalierbarkeit:** Die Middleware verwaltet Roboter unabhängig voneinander. Solange Hardware-Ressourcen ausreichen, ist horizontale Skalierung durch zusätzliche Roboter möglich.
- **Verteilungstransparenzen:** Durch das 3-Schichtenmodell und das MMI über die GUI werden die Verteilungstransparenzen teilweise eingehalten. Die Architektur versteckt die konkrete Lage und Ansteuerung der Roboter (Ortstransparenz, Zugriffstransparenz) vor dem Nutzer. Die GUI zeigt nur logische Namen an, nicht IP-Adressen oder spezifische Schnittstellen.

Schwächen

- Netzwerk kann durch UDP-Anfragen "lahmgelegt werden", Watchdog an Middleware und am Roboterarm darf also nicht auf Antwort warten
- Single Point of Failure. SE-Ziel Zuverlässigkeit durch Absturz von Middleware möglicherweise nicht gegeben.
- Ziel Leistung: Leistung des Systems durch Middleware abhängig. (Befehle pro Sekunde)
- Fehlertransparenz: GUI zeigt nicht verfügbaren Roboter nicht mehr an.
- Skalierbarkeitstransparenz: Neue Roboter könnten durch Watchdog zu Leistungseinbrüchen des Systems durch Middleware führen
- Lokalitätstransparenz: Ist durch den eindeutigen Namen nicht gegeben und durch Hinzufügen bzw. Verschwinden auf der GUI nicht gegeben.

5 Bausteinsicht

6 Laufzeitsicht

7 Verteilungssicht

8 Konzepte

8.1 Fachliche Strukturen (Domäne)

Fachliche Modelle, Domänenmodelle, Business-Modelle – sie alle beschreiben Strukturen der reinen Fachlichkeit, also ohne Bezug zur Implementierungs- oder Lösungstechnologie. Oftmals tauchen Teile solcher fachlichen Modelle an vielen Stellen in der Architektur, insbesondere der Bausteinsicht, wieder auf. Das "Domain-Driven-Designöder die uralte ëssentielle Systemanalyse"können Ihnen hierbei helfen.

Mit Absicht stellen wir diesen Abschnitt an den Anfang der übergreifenden Konzepte.

8.2 Typische Muster und Strukturen

Oftmals tauchen einige typische Lösungsstrukturen oder Grundmuster an mehren Stellen der Architektur auf. Beispiele dafür sind die Abhängigkeiten zwischen Persistenzschicht, Applikation sowie die Anbindung grafischer Oberflächen an die Fach- oder Domänenobjekte. Solche wiederkehrenden Strukturen beschreiben Sie möglichst nur ein einziges Mal, um Redundanzen zu vermeiden. Dieser Abschnitt erfüllt genau diesen Zweck.

8.3 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.

8.4 Ausnahme- und Fehlerbehandlung

Wie werden Programmfehler und Ausnahmen systematisch und konsistent behandelt? Wie kann das System nach einem Fehler wieder in einen konsistenten Zustand gelangen? Geschieht dies automatisch oder ist manueller Eingriff erforderlich? Dieser Aspekt hat mit Logging, Protokollierung und Tracing zu tun. Welche Art Ausnahmen und Fehler behandelt ihr System? Welche Art Ausnahmen werden an welche Außenschnittstelle weitergeleitet und welche Ausnahmen behandelt das System komplett intern? Wie nutzen Sie die Exception-Handling Mechanismen ihrer Programmiersprache? Verwenden Sie checked- oder unchecked-Exceptions?

8.5 Batchverarbeitung

Batchverarbeitung sequentielle Verarbeitung einer i.d.R. vorab festgelegten Menge an Daten oder Aufgaben.

8.6 Bedienoberfläche

IT-Systeme, die von (menschlichen) Benutzern interaktiv genutzt werden, benötigen eine Benutzungsoberfläche. Das können sowohl grafische als auch textuelle Oberflächen sein.

8.7 Ergonomie

Ergonomie von IT-Systemen bedeutet die Verbesserung (Optimierung) deren Benutzbarkeit aufgrund objektiver und subjektiver Faktoren. Im wesentlichen zählen zu ergonomischen Faktoren die Benutzungsoberfläche, die Reaktivität (gefühlte Performance) sowie die Verfügbarkeit und Robustheit eines Systems.

8.8 Geschäftsregeln

Wie behandeln Sie Geschäftslogik oder Geschäftsregeln? Implementieren die beteiligten Fachklassen ihre Logik selbst, oder liegt die Logik in der Verantwortung einer zentralen Komponente? Setzen Sie eine Regelmaschine (rule-engine) zur Interpretation von Geschäftsregeln ein (Produktionsregelsysteme, forward- oder backward-chaining)?

8.9 Hochverfügbarkeit

Wie erreichen Sie hohe Verfügbarkeit des Systems? Legen Sie Teile redundant aus? Verteilen Sie das System auf unterschiedliche Rechner oder Rechenzentren? Betreiben Sie Standby-Systeme? Könnte in Zusammenhang zu Clusterung stehen.

8.10 Internationalisierung

Unterstützung für den Einsatz von Systemen in unterschiedlichen Ländern, Anpassung der Systeme an länderspezifische Merkmale. Bei der Internationalisierung (aufgrund der 18 Buchstaben zwischen I und n des englischen Internationalisation auch i18n genannt) geht es neben der Übersetzung von Aus- oder EIngabetexten auch um verwendete Zeichensätze, Orientierung von Schriften am Bildschirm und andere (äußerliche) Aspekte.

8.11 Kommunikation, Integration

Kommunikation: Übertragung von Daten zwischen System-Komponenten. Bezieht sich auf Kommunikation innerhalb eines Prozesses oder Adressraumes, zwischen un-

terschiedlichen Prozessen oder auch zwischen unterschiedlichen Rechnersystemen. Integration: Einbindung bestehender Systeme (in einen neuen Kontext). Auch bekannt als: (Legacy) Wrapper, Gateway, Enterprise Application Integration (EAI).

8.12 Konfiguration

Die Flexibilität von IT-Systemem wird unter anderem durch ihre Konfigurierbarkeit beeinflusst, die Möglichkeit, manche Entscheidungen hinsichtlich der Systemnutzung erst spät zu treffen. Konfiguration kann zu folgenden Zeitpunkten erfolgen: Während der Programmierung: Dabei werden beispielsweise Server-, Datei- oder Verzeichnisnamen direkt ("hart") in den Programmcode aufgenommen. Während des Deployments oder der Installation: Hier werden Konfigurationsinformationen für eine bestimmte Installation angegeben, etwa der Installationspfad. Beim Systemstart: Hier werden Informationen vor oder beim Programmstart dynamisch gelesen. Während des Programmablaufs: Konfigurationsinformation wird zur Programmlaufzeit erfragt oder gelesen.

8.13 Logging, Protokollierung

Es gibt zwei Ausprägungen der Protokollierung, das Logging und das Tracing . Bei beiden werden Funktions- oder Methodenaufrufe in das Programm aufgenommen, die zur Laufzeit über den Status des Programms Auskunft geben. In der Praxis gibt es zwischen Logging und Tracing allerdings sehr wohl Unterschiede:

- Logging kann fachliche oder technische Protokollierung sein, oder eine beliebige Kombination von beidem.
- Fachliche Protokolle werden gewöhnlich anwenderspezifisch aufbereitet und übersetzt. Sie dienen Endbenutzern, Administratoren oder Betreibern von Softwaresystemen und liefern Informationen über die vom Programm abgewickelten Geschäftsprozesse.
- Technische Protokolle sind Informationen für Betreiber oder Entwickler. Sie dienen der Fehlersuche sowie der Systemoptimierung.
- Tracing soll Debugging -Information für Entwickler oder Supportmitarbeiter liefern. Es dient primär zur Fehlersuche und -analyse.

8.14 Management und Administrierbarkeit

Größere IT-Systeme laufen häufig in kontrollierten Ablaufumgebungen (Rechenzentren) unter der Kontrolle von Operatoren oder Administratoren ab. Diese Stakeholder benötigen einerseits spezifische Informationen über den Zustand der Programme zur Laufzeit, andererseits auch spezielle Eingriffs- oder Konfigurationsmöglichkeiten.

8.15 Migration

Für manche Systeme gibt es existierende Altsysteme, die durch die neuen Systeme abgelöst werden sollen. Denken Sie als Architekt rechtzeitig auch an alle organisatorischen und technischen Aspekte, die zur Einführung oder Migration der Architektur beachtet werden müssen. Beispiele:

- Konzept, Vorgehensweise oder Werkzeuge zur Datenübernahme und initialen Befüllung mit Daten
- Konzept zur Systemeinführung oder zeitweiliger Parallelbetrieb von Alt- und Neusystem

Müssen Sie bestehende Daten migrieren? Wie führen Sie die benötigten syntaktischen oder semantischern Transformationen durch?

8.16 Parallelisierung / Nebenläufigkeit

Programme können in parallelen Prozessen oder Threads ablaufen - was die Notwendigkeit von Synchronisationspunkten mit sich bringt. Die Grundlagen dieses Aspekten legt die Parallelverarbeitung. Für die Architektur und Implementierung nebenläufiger Systeme sind viele technische Detailaspekte zu berücksichtigen (Adressräume, Arten von Synchronisationsmechanismen (Guards, Wächter, Semaphore), Prozesse und Threads, Parallelität im Betriebssystem, Parallelität in virtuellen Maschinen und andere).

8.17 Persistenz

Persistenz (Dauerhaftigkeit, Beständigkeit) bedeutet, Daten aus dem (flüchtigen) Hauptspeicher auf ein beständiges Medium (und wieder zurück) zu bringen. Einige der Daten, die ein Software-System bearbeitet, müssen dauerhaft auf einem Speichermedium gespeichert oder von solchen Medien gelesen werden:

- Flüchtige Speichermedien (Hauptspeicher oder Cache) sind technisch nicht für dauerhafte Speicherung ausgelegt. Daten gehen verloren, wenn die entsprechende Hardware ausgeschaltet oder heruntergefahren wird.
- Die Menge der von kommerziellen Software-Systemen bearbeiteten Daten übersteigt üblicherweise die Kapazität des Hauptspeichers.
- Auf Festplatten, optischen Speichermedien oder Bändern sind oftmals große Mengen von Unternehmensdaten vorhanden, die eine beträchtliche Investition darstellen.

Persistenz ist ein technisch bedingtes Thema und trägt nichts zur eigentlichen Fachlichkeit eines Systems bei. Dennoch müssen Sie sich als Architekt mit dem Thema auseinander setzen, denn ein erheblicher Teil aller Software-Systeme benötigt einen effizienten Zugriff auf persistent gespeicherte Daten. Hierzu gehören praktisch sämtliche kommerziellen und viele technischen Systeme. Eingebettete Systeme (embedded systems) gehorchen jedoch oft anderen Regeln hinsichtlich ihrer Datenverwaltung.

8.18 Plausibilisierung und Validierung

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

8.19 Sessionbehandlung

Eine Session, auch genannt Sitzung, bezeichnet eine stehende Verbindung eines Clients mit einem Server. Den Zustand dieser Sitzung gilt es zu erhalten, was insbesondere bei der Nutzung zustandsloser Protokolle (etwa HTTP) wichtige Bedeutung hat. Sessionbehandlung stellt für Intra- und Internetsysteme eine kritische Herausforderung dar und beeinflusst häufig die Performance eines Systems.

8.20 Sicherheit

Die Sicherheit von IT-Systemen befasst sich mit Mechanismen zur Gewährleistung von Datensicherheit und Datenschutz sowie Verhinderung von Datenmissbrauch. Typische Fragestellungen sind:

- Wie können Daten auf dem Transport (beispielsweise über offene Netze wie das Internet) vor Missbrauch geschützt werden?
- Wie können Kommunikationspartner sich gegenseitig vertrauen?
- Wie können sich Kommunikationspartner eindeutig erkennen und vor falschen Kommunikationspartner schützen?
- Wie können Kommunikationspartner die Herkunft von Daten für sich beanspruchen (oder die Echtheit von Daten bestätigen)?

Das Thema IT-Sicherheit hat häufig Berührung zu juristischen Aspekten, teilweise sogar zu internationalem Recht.

8.21 Skalierung / Clusterung

Wie gestalten Sie Ihr System „wachstumsfähig“, so daß auch bei steigender Last oder steigenden Benutzerzahlen die Antwortzeiten und/oder Durchsatz erhalten bleiben?

8.22 Transaktionsbehandlung

Transaktionen sind Arbeitsschritte oder Abläufe, die entweder alle gemeinsam oder garnicht durchgeführt werden. Der Begriff stammt aus den Datenbanken - wichtiges Stichwort hier sind ACID-Transaktionen (atomar, consistent, isolated, durable). Im Bereich von NoSQL-Datenbanken gelten andere Kriterien.

8.23 Verteilung

Verteilung: Entwurf von Software-Systemen, deren Bestandteile auf unterschiedlichen und eventuell physikalisch getrennten Rechnersystemen ablaufen. Zur Verteilung gehören Dinge wie der Aufruf entfernter Methoden (remote procedure call, RPC), die Übertragung von Daten oder Dokumenten an verteilte Kommunikationspartner, die Wahl passender Interaktionsstile oder Nachrichtenaustauschmuster (etwa: synchron / asynchron, publish- subsribe, peer-to- peer).

9 Entwurfsentscheidungen

10 Qualitätsszenarien

10.1 Bewertungsszenarien

10.2 Qualitätsbaum

11 Risiken

Beispieleintrag für ein Glossarverweis: Architekturdokumentation.