Graphische Testfallmodellierung mit UML – Expertenwissen in Bildern?

Dipl.-Ing. Volker Knollmann

Deutsches Zentrum
DLR für Luft- und Raumfahrt e.V.
in der Helmholtz-Gemeinschaft

Motivation

Verbesserungspotential bei Testfallbeschreibungen

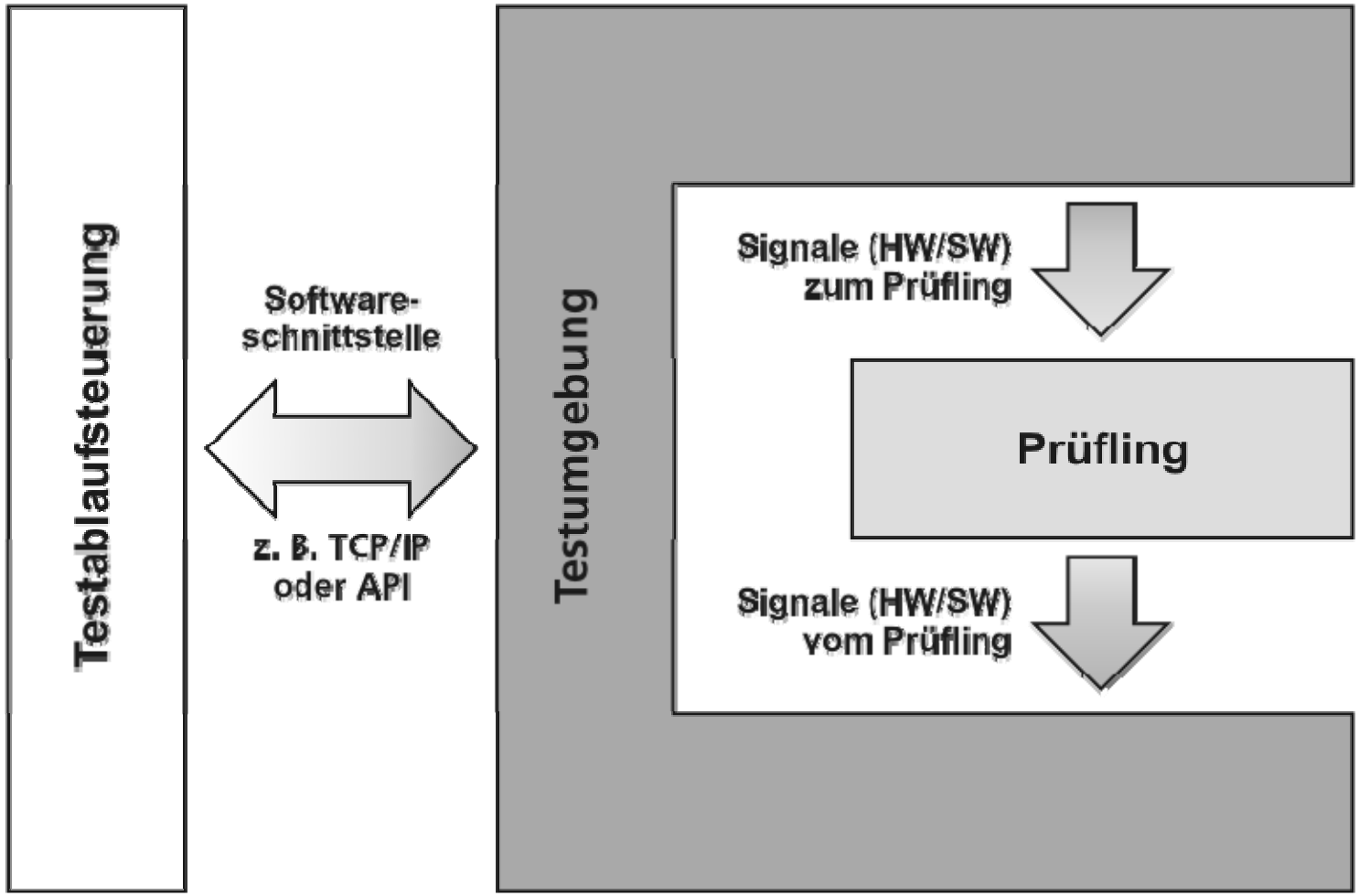
- **Beispiel: Betrieb des Eisenbahnsimulationslabors RailSiTe®**
  - interne Tests: notwendig bei Updates / Laborvalidierung
  - externe Tests: Überprüfung von Kundengeräten
- **Beispiel: Tests für das European Train Control System (ETCS)**
  - Testsequenzbeschreibung als Worddatei veröffentlicht
  - Geschwindigkeitsprofil als Bitmap verfügbar
- **Beispiel: Simulation bahnbetrieblicher Szenarien**
  - Betriebsregeln nur in “Prosa” definiert
  - Validierung neuer Eisenbahnleit- und -sicherungssysteme aufwändig

Gliederung des Vortrags

- Anforderungen / Komponenten von Testfallbeschreibungen
- Das „UML 2 Testing Profile“ (U2TP)
- Anwendung des U2TP im Rahmen einer Fallstudie
- Ausblick auf weitere Arbeiten
- Fazit

Abgrenzung des Kontextes

Hier nur Betrachtung von Black-Box-Tests



Anforderungen an Testfallbeschreibungen

Editierbar, automatisierbar, archivierbar, rückverfolgbar

Beschreibung muss verständlich und leicht editierbar sein

Verwendung von UML
(Unified Modeling Language)

Komponenten einer Testbeschreibung

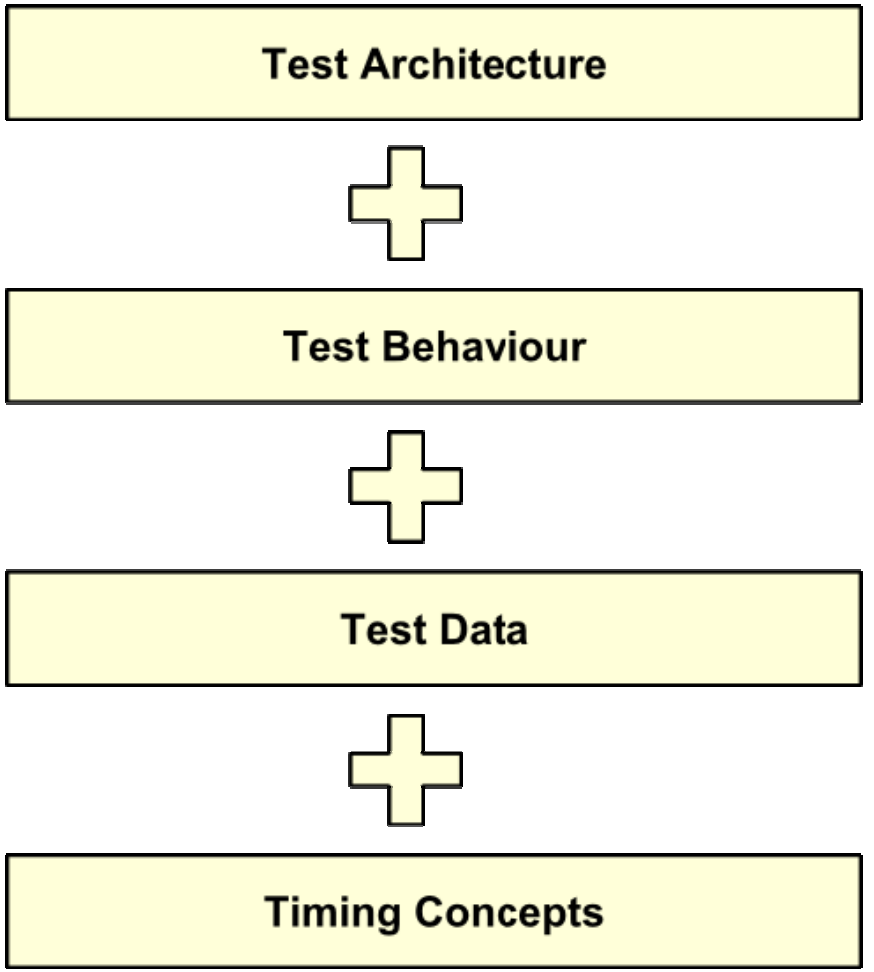
Struktur, Verhalten, Daten und zeitliche Bedingungen

- **Statische Anteile (Struktur):**
  - Aufbau und Konfiguration der Testumgebung
  - Verwendete Schnittstellen
- **Dynamische Anteile (Verhalten):**
  - Testschritte
  - Sollergebnisse, Reaktion bei Abweichung vom Sollverhalten
  - Ableitung der Testbewertung (bestanden / nicht bestanden)
- **Daten und zeitliche Randbedingungen:**
  - Stimuli, Telegramme, Datenpakete
  - Timer, Timeouts

Das UML 2 Testing Profile (U2TP)

OMG[1]-Standard seit Juli 2005

- Definiert Symbole, Stereotypen und Tags für Testfallbeschr.
- Baut auf UML 2 auf
- Schafft semantische Eindeutigkeit der Beschreibung
- Deckt die vorgenannten Modellaspekte ab
- Anwendbar auf allen Testebenen (z. B. Modultest, Systemtest, …)
- Werkzeug- und plattformneutral
- Nicht domänenspezifisch



[1] Object Management Group

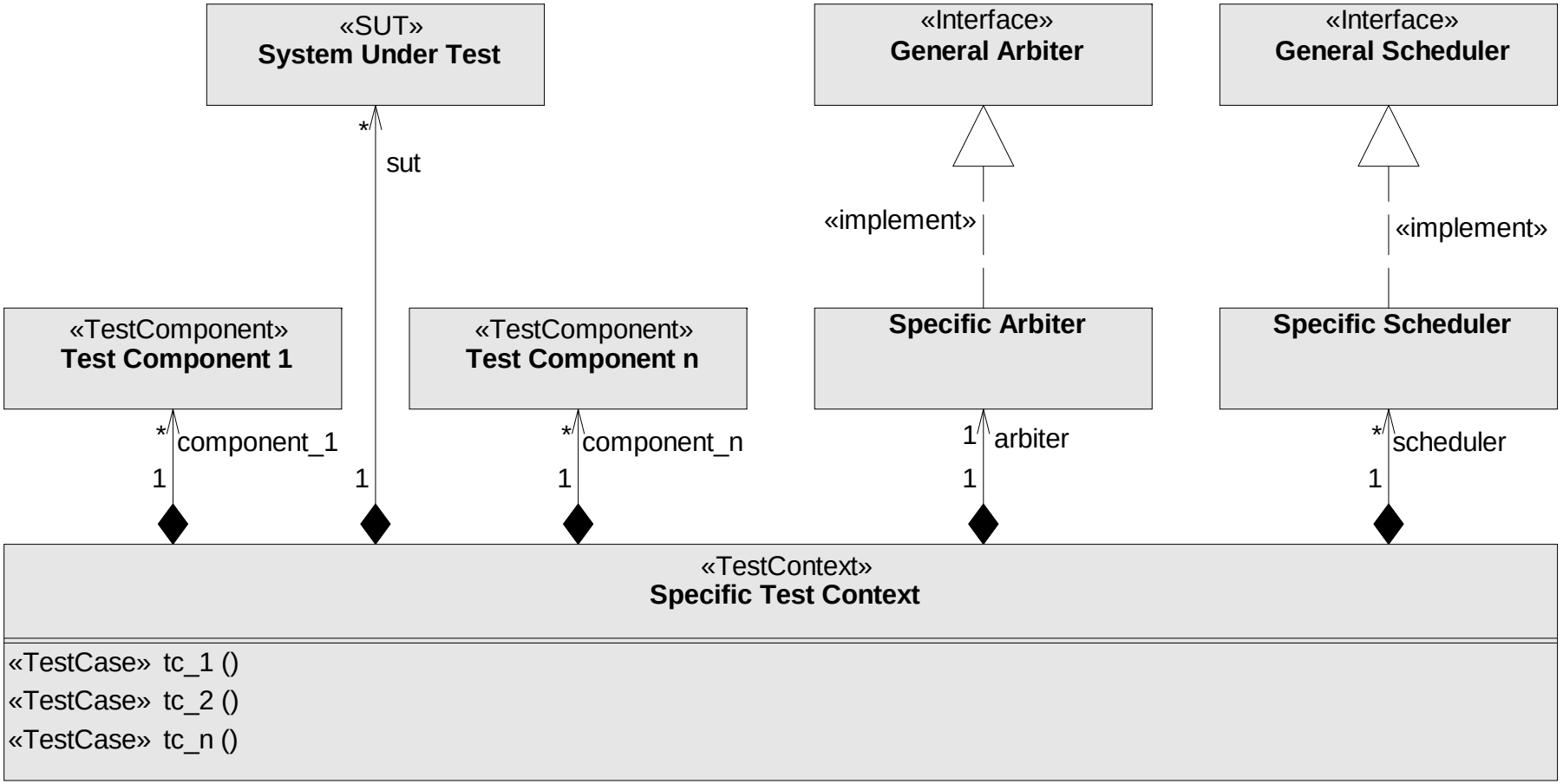
Komponenten des Moduls „Test Architecture“

Testaufbau als Klassen- und Strukturdiagramm

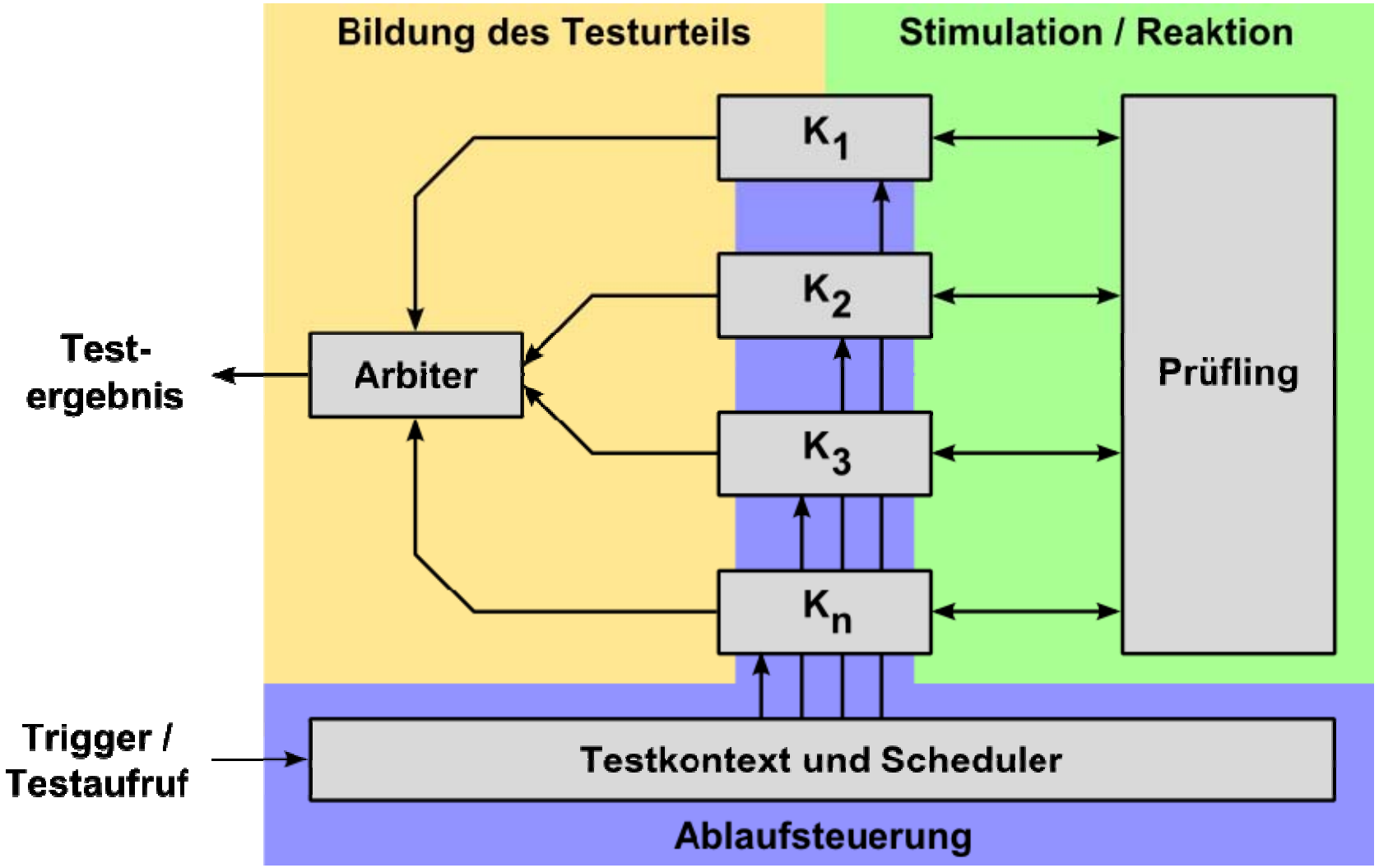
- **Prüfling („System under Test“, SUT)**
- **Testkomponenten:**
  - Sind Teil der Testumgebung
  - Interagieren mit dem SUT (Stimuli / Reaktionen)
- **Arbiter:**
  - Empfängt und bewertet Rückmeldungen der Testkomponenten
  - Fällt Gesamturteil über Testergebnis
- **Scheduler:**
  - Steuerung des zeitlichen Testablaufs
  - Starten / Stoppen einzelner Testkomponenten

Zusammenspiel der Architektur-Komponenten

Der Testkontext integriert Komponenten zum Testaufbau



Daten- und Steuersignalfluss in der Testumgebung



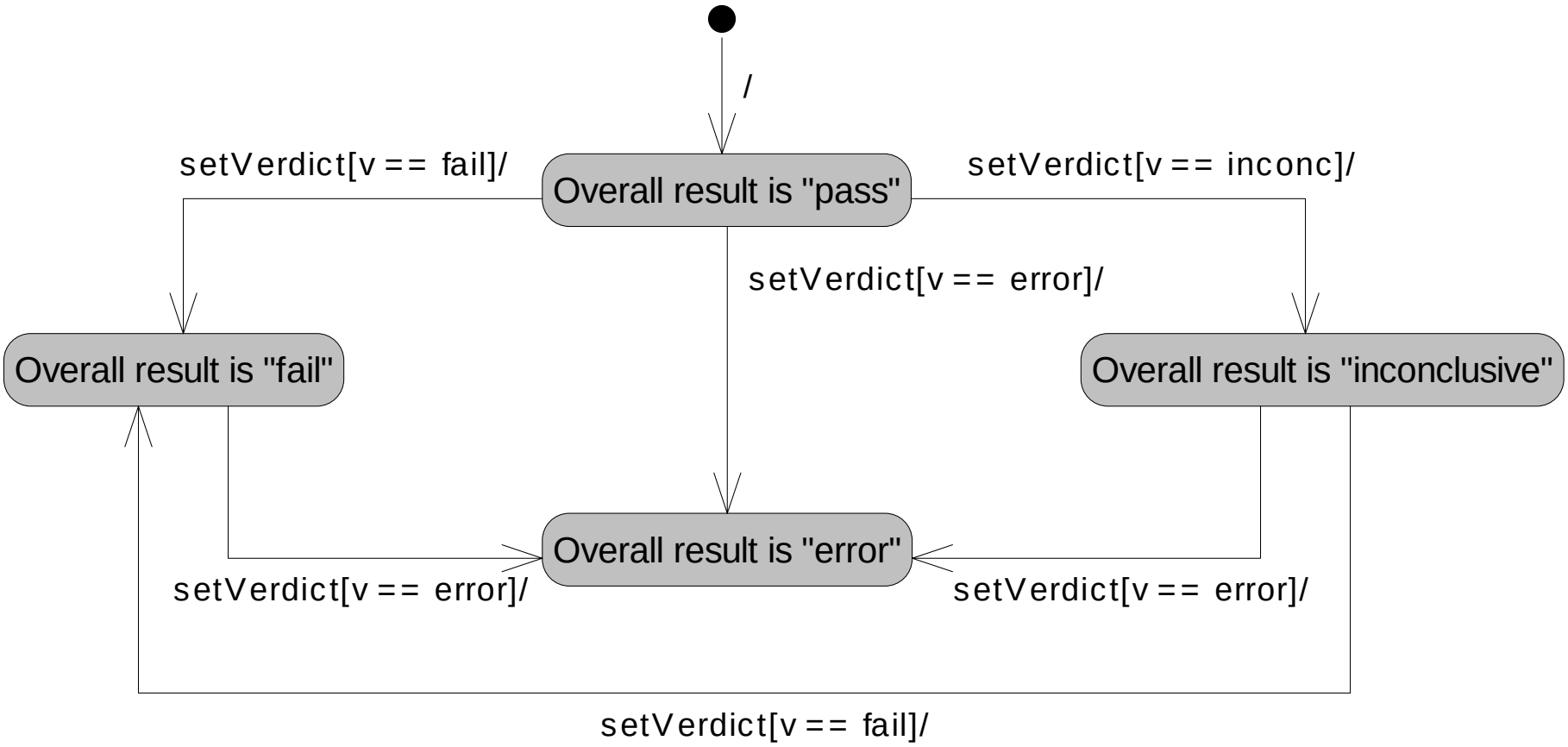
Beschreibung des Testverhaltens

Nutzung von Sequenz-, Aktivitäts- und Zustandsdiagr.

- Nutzung der gängigen UML-Verhaltensdiagramme
- Testablauf beispielsweise als Sequenzdiagramm
- Einführung sogenannter „defaults“:
  - Definieren Reaktion bei Abweichung vom Sollverhalten
  - Entspricht dem Aufruf einer Fehlerbehandlungsroutine
  - Fehlerbehandlung wird separat modelliert (→ Übersichtlichkeit!)
  - Mögliche Reaktionen: Wiederholung, Ignorieren, Abbruch
- Verhaltensdefinition von Scheduler und Arbiter

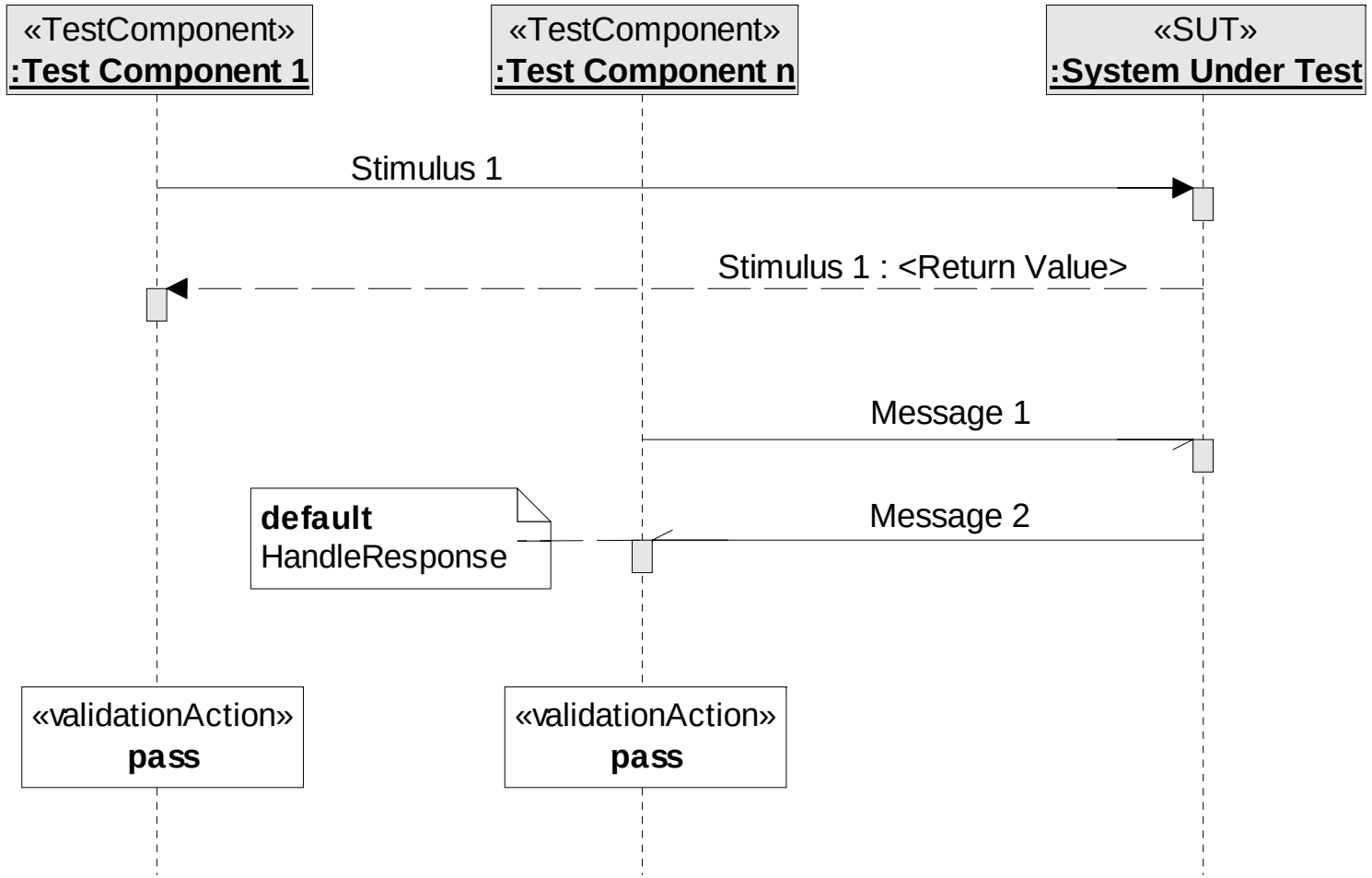
Beispiel einer einfachen Arbiter-Implementierung

Hier: fällt eine Worst-Case-Entscheidung



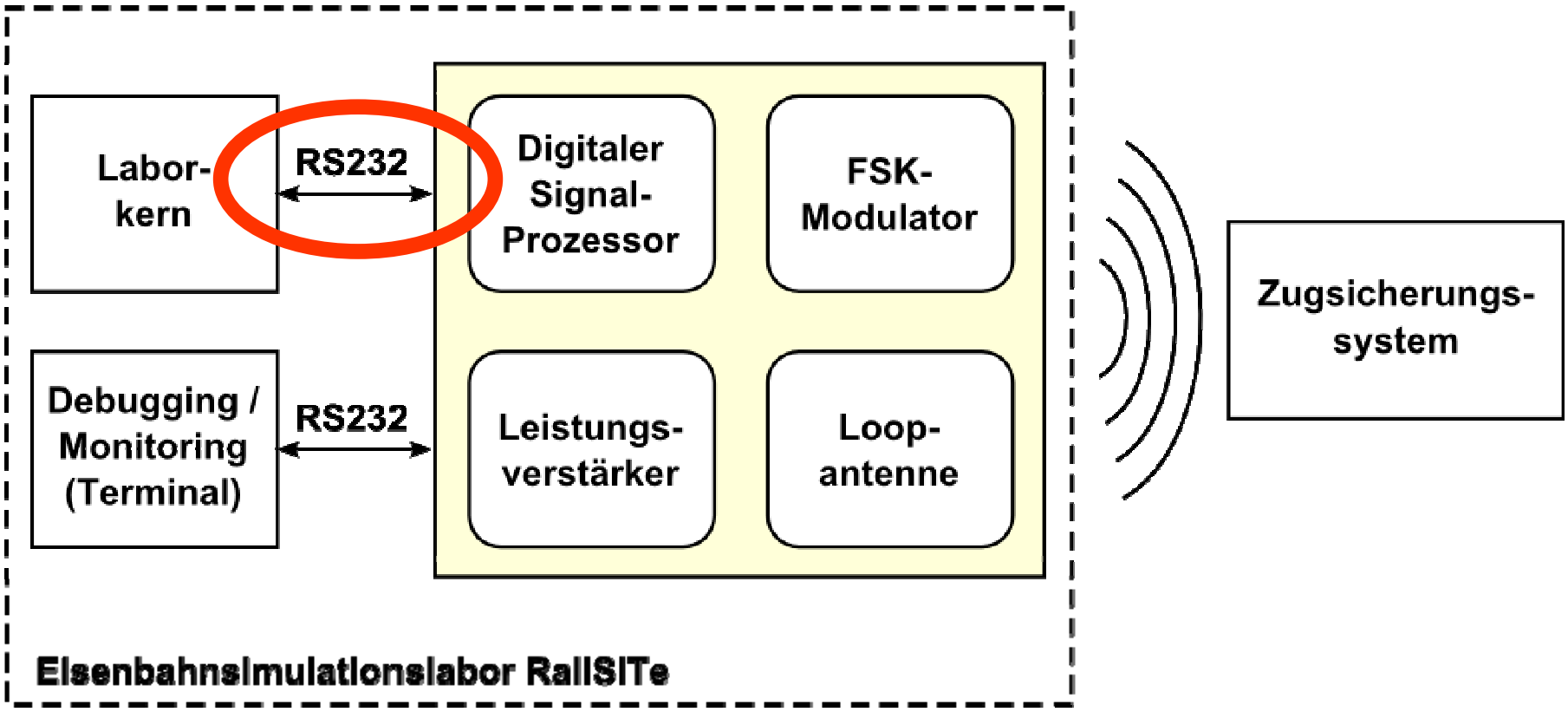
Beispiel einer Testablaufbeschreibung

Austausch synchroner und asynchroner Nachrichten



Gegenstand der Fallstudie

System zur Erzeugung von Transpondersignalen



Gegenstand der Fallstudie

Eigenschaften des untersuchten RS232-Protokolls

- **Transportierte Inhalte:**
  - Die per Transpondersignal zu übertragenden Daten
  - Zeitprofil für die Datenübertragung
  - Steuerkommandos (Reset, Statusabfragen, etc)
- **Protokoll:**
  - Paketorientiertes Protokoll
  - Zweischichtig: Link-Layer- und Applikationsebene
- **Aufgaben des Link-Layers:**
  - Verwaltung von Sequenznummern
  - Vollständigkeitsprüfung der Pakete (Längeninformation)
  - Prüfsummenkontrolle

Vorgehen im Rahmen der Fallstudie

Testfallableitung, Testdatenerzeugung, Modellierung

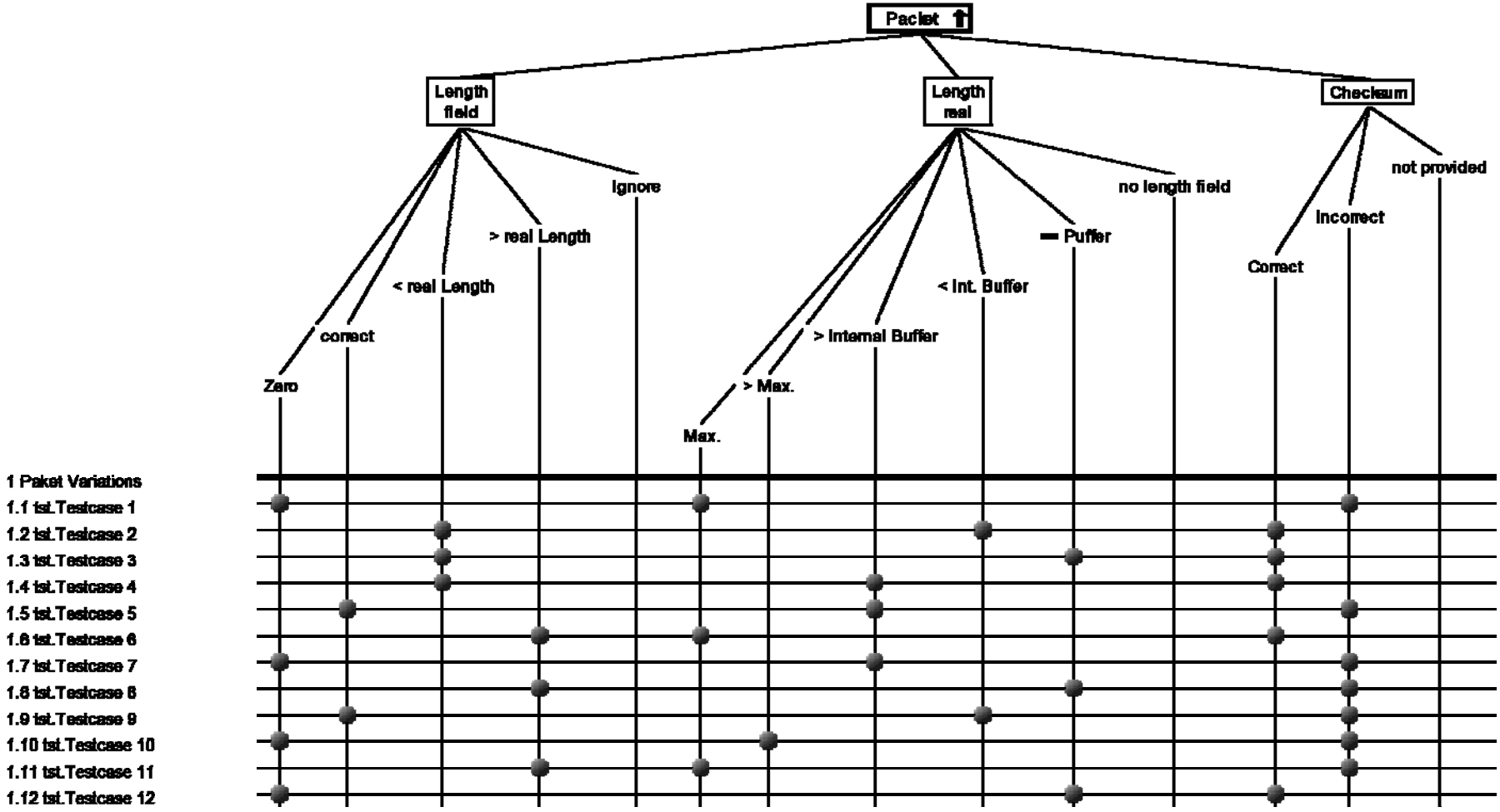
Methodische Ableitung von Testfällen mit Klassifikationsbäumen

Erzeugung zugehöriger Datenpakete (Testdaten)

Modellierung der sog. „Datenpools“ und der Testumgebung in UML (Prüflingsmodell lag bereits in UML vor)

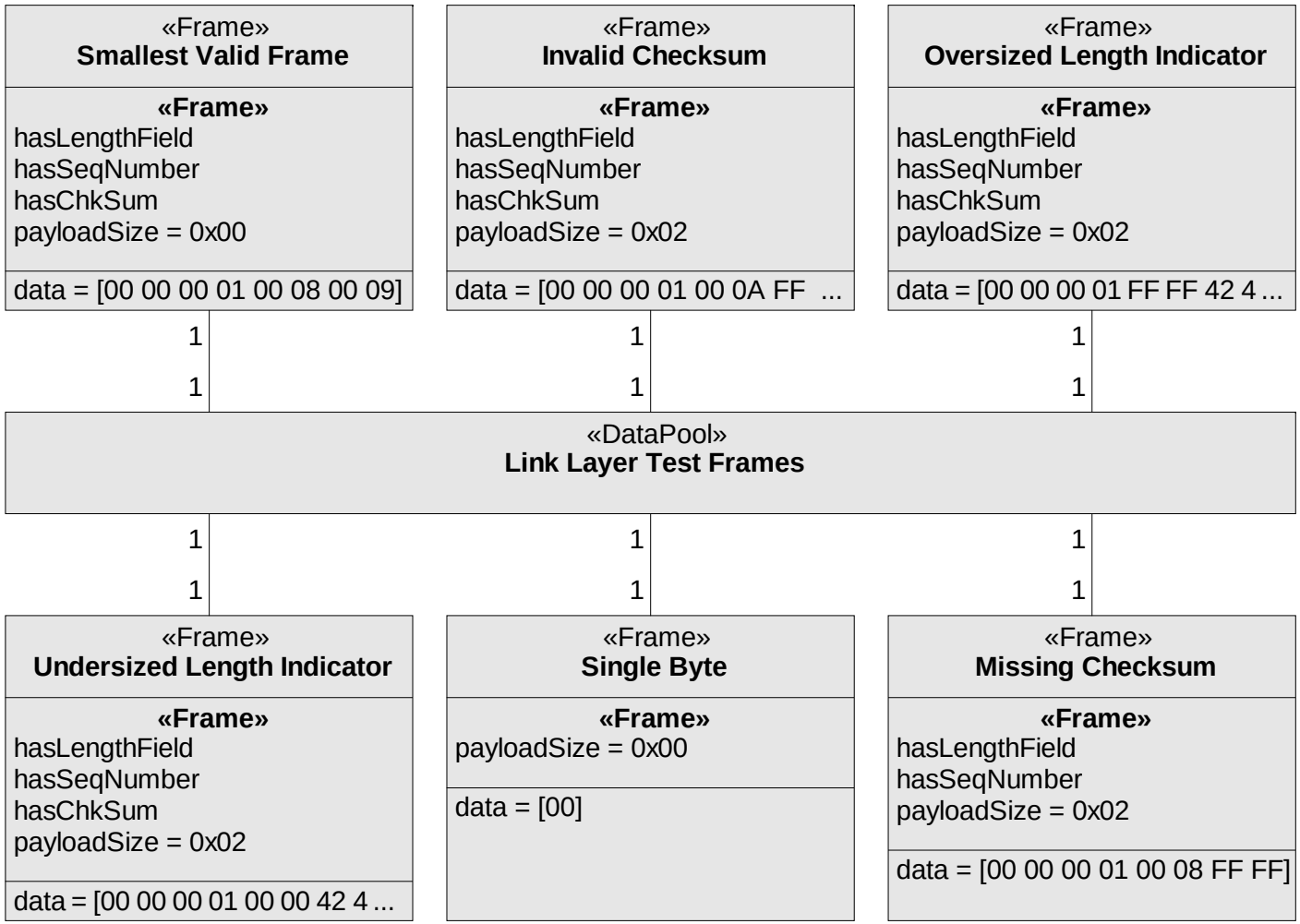
Testfallableitung mit Klassifikationsbäumen

Basiert auf Partitionierung des Zustandsraumes



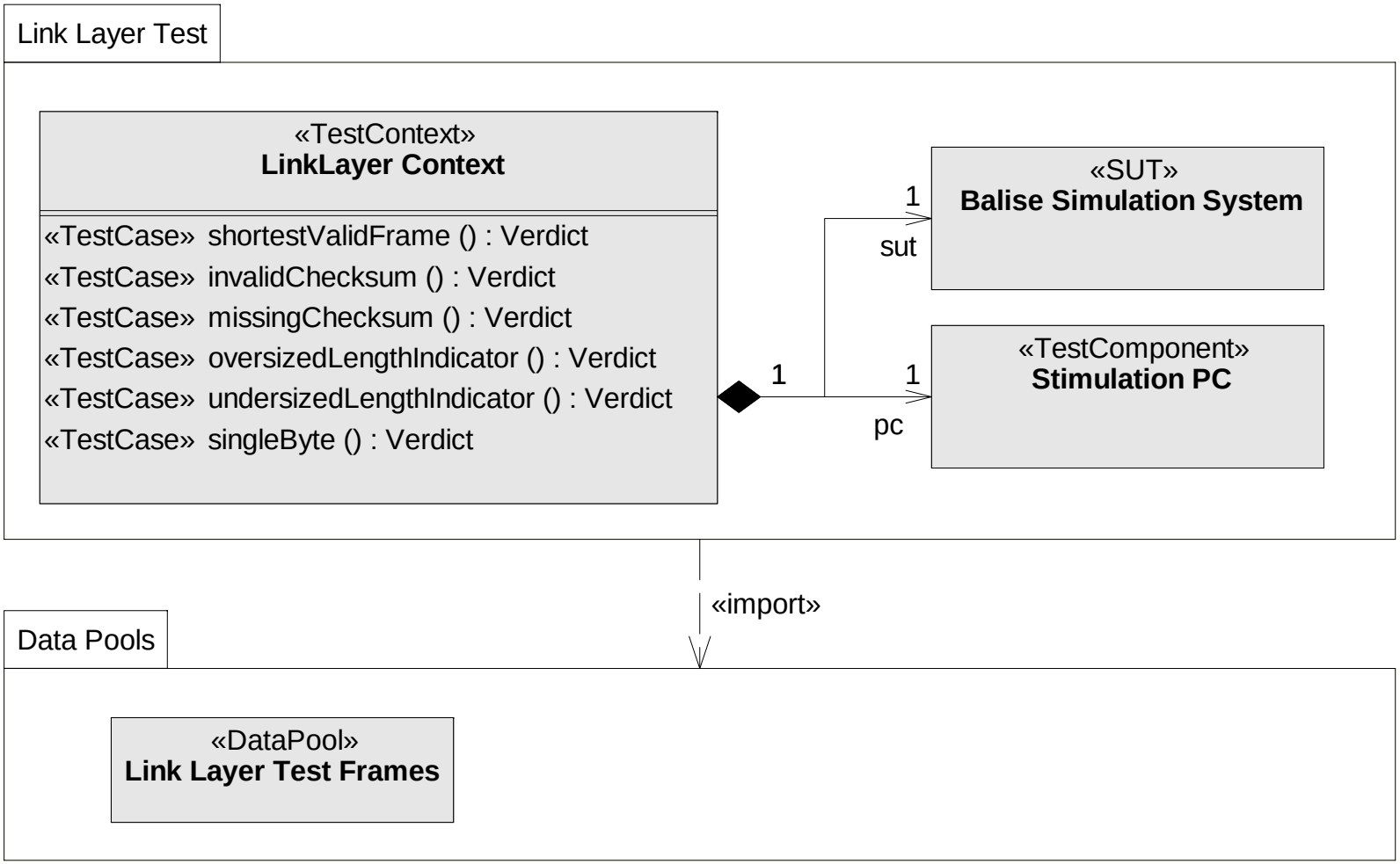
Anordnen der Testdaten in einem Datenpool

Insgesamt wurden 41 Testfälle erzeugt



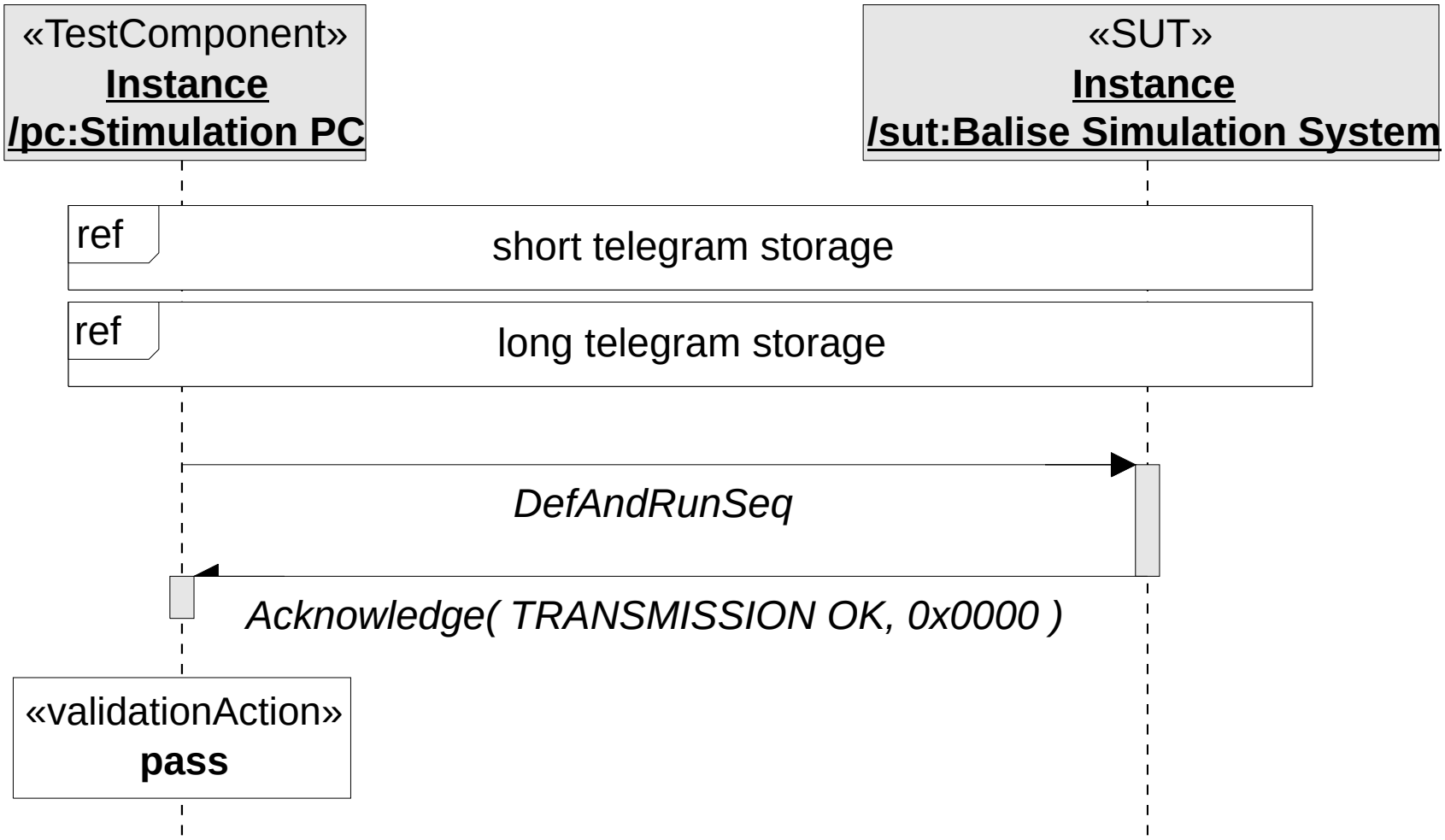
Integration aller Komponenten zum Testkontext

Hier nur oberste Ebene dargestellt; Details sind hinterlegt



Beispiel einer Testsequenz

Wiederverwendung anderer Sequenzen ist möglich



Ergebnisse und Bewertung der Fallstudie

Die Vorteile überwiegen

- **Vorteile:**
  - Modularer Aufbau, Wiederverwendung von Komponenten
  - Übersichtliche, formale Darstellung; gute Editier- und Wartbarkeit
  - Erfassung des Testaufbaus (wird häufig vergessen)
  - Nahtlose Integration mit Systemmodell möglich
- **Nachteile:**
  - Einarbeitung in UML und U2TP erforderlich
  - Viele Diagramme, daher saubere Strukturierung notwendig
  - Für komfortables Arbeiten ist gutes UML-Tool unerlässlich
  - Initial höherer Aufwand als bei traditionellen Methoden

Ausblick: Übergang zur Testautomatisierung

Verschiedene Ansätze sind denkbar

- **Konvertierung UML → XMI[1] → Testskript:**
  - Übersetzung in ein ausführbares Skript (z. B. Python oder Ruby)
  - Ausführen des Skriptes in der Testumgebung

- **Direktes Ausführen von XMI in der Testumgebung:**
  - Erstellung eines entsprechenden „XMI-Interpreters“
  - Positiv: Erstellung eines ausführbaren Skriptes entfällt

- **Konvertierung in alternative Datenformate:**
  - Nutzung von XSLT[2] zur Konvertierung in andere XML[3]-Formate
  - Alternativ: Übergang auf TTCN-3[4]
  - Mapping U2TP → TTCN-3 existiert bereits

[1] XML Metadata Interchange; [2] eXtensible Stylesheet Language Transformation;
[3] eXtensible Markup Language; [4] Testing and Test Control Notation, Version 3

Fazit
Expertenwissen in Bildern?

- U2TP-Darstellung von Testfällen ist übersichtlich, einheitlich, formal
- Nur für Black-Box-Tests vorgesehen
- Geeignete Grundlage für Testautomatisierung
- Aufbau von Bibliotheken möglich
- Konservierung, nicht Ersatz des Expertenwissens durch „Bilder“
- Entlastung des Testern von Routineaufgaben
- Kein Ersatz für die „Kreativität“ des Testers