Institut für Architektur von Anwendungssystemen

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

# Choreographiesprachen mit Datenmodellierungsfähigkeiten – eine Übersicht

Nico Lässig

| | |
|---|---|
| **Studiengang:** | Informatik |
| **Prüfer/in:** | Prof. Dr. Dr. h.c. Frank Leymann |
| **Betreuer/in:** | Dr. Oliver Kopp,<br>Dipl.-Inf. Michael Hahn |
| **Beginn am:** | 14. Dezember 2016 |
| **Beendet am:** | 14. Juni 2017 |
| **CR-Nummer:** | C.2.4, D.2.2, D.2.11, H.4.1 |

# Kurzfassung

In organisationsübergreifenden Geschäftsprozessen ist die Kommunikation zwischen den teilhabenden Diensten von zentraler Bedeutung. Choreographiesprachen liefern eine globale Sichtweise auf das Zusammenwirken der verschiedenen Dienste in solchen Geschäftsprozessen und legen den Fokus eben auf diese Kommunikation.

Für die Modellierung von Choreographien existieren bereits eine Vielzahl an Sprachen, die jedoch aufgrund der unterschiedlichen Anforderungen oft unterschiedliche Modellierungskonstrukte und Eigenschaften haben. In dieser Arbeit sollen die existierenden Sprachen gefunden und im Anschluss miteinander verglichen und analysiert werden. Dafür werden Kriterien definiert, welche Choreographiesprachen erfüllen sollten. Es wird dabei besonders auf Datenmodellierungsaspekte eingegangen, da der Austausch von Daten ein wichtiger Faktor für die Interaktionen zwischen den verschiedenen Diensten ist.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Verzeichnis der Listings

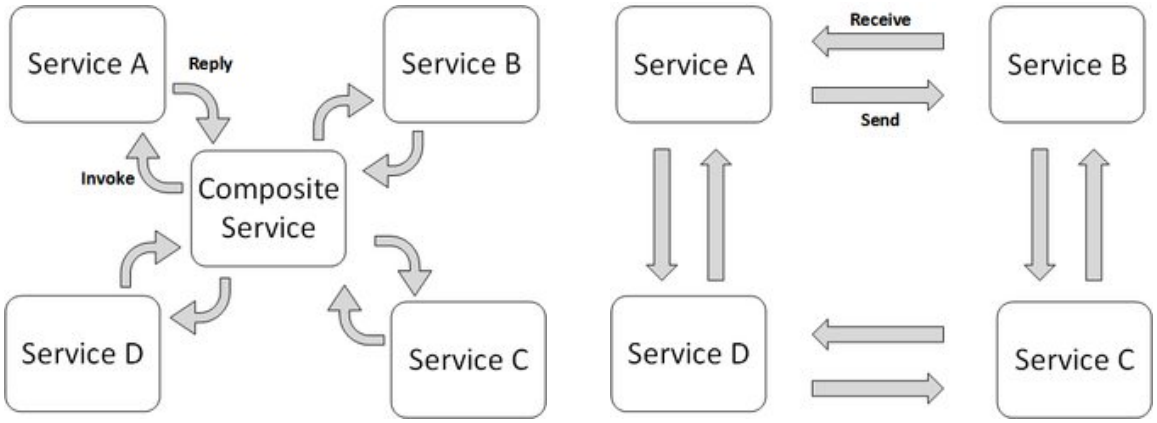
# 1 Einleitung

In vielen Arbeitsprozessen sind mehrere Organisationen involviert, welche für unterschiedliche Aufgaben zuständig sind. Diese Arbeitsprozesse nennt man auch organisationsübergreifende oder unternehmensübergreifende Geschäftsprozesse. Für die Modellierung von solchen Prozessen wird zwischen zwei Arten unterschieden, *Orchestrierungen* und *Choreographien*, deren Konzepte in Abbildung 1.1 dargestellt sind. In *Orchestrierungen* gibt es eine zentrale Leitung, welche für die Koordination zwischen den Diensten zuständig ist. Deshalb liegt der Vordergrund bei der Modellierung von Orchestrierungen bei dem organisationsinternen Ablauf eines Prozesses [Kop16]. Hier geht es also hauptsächlich um das „Wie werden die einzelnen Aufgaben erfüllt?“. Orchestrierungen sind außerdem meist ausführbare Geschäftsprozesse. Die Sicht der einzelnen Dienste ist dabei jedoch beschränkt. *Choreographien* hingegen sind nicht zwingend ausführbar, aber bieten eine globale Sichtweise auf das Zusammenwirken von Diensten in organisationsübergreifenden Geschäftsprozessen [DKB08]. Hier wird der Wert auf die Interaktion zwischen den einzelnen Diensten gelegt, also „Wie wird zwischen den Diensten miteinander kommuniziert?“. Hier gibt es keine zentrale Leitung [Kop16].

Dabei gibt es nochmal Unterteilungen von Choreographien für spezifische Bereiche, wie *Service Choreographien*, *“Business-to-Business integration“ (kurz: B2Bi) Choreographien*, *Konzeptionelle Choreographien* [Sch11], *Prozess Choreographien* [JHKK04] und *Artefakt-zentrierte Choreographien* [FDVV11]. Sie können auch durch die Art des Modells der Choreographiesprache unterschieden werden (Verbindungsmodell vs. Interaktionsmodell). In dieser Bachelorarbeit wird allerdings kein spezieller Bereich untersucht, sondern die Übermenge „Choreographie“ betrachtet.

Diese Bachelorarbeit beschäftigt sich mit Sprachen, welche zur Modellierung von Choreographien geeignet sind. Für die Modellierung von Choreographien wurden über die Jahre bereits eine Vielzahl an Sprachen entwickelt, die jedoch oft unterschiedliche Modellierungskonstrukte und Eigenschaften haben [Kop16]. Ein Grund dafür ist, dass jeder Entwickler solch einer Sprache andere Anforderungen hat, was für Kriterien eine solche Sprache erfüllen soll. Es gibt keine allgemeingültige Richtlinie dafür, welche Kritieren Choreographiesprachen erfüllen müssen. Diese werden jeweils selbst festgelegt. Dadurch sind die verschiedenen Modellierungsmöglichkeiten, welche diese Choreographiesprachen liefern sehr vielfältig und von Sprache zu Sprache unterschiedlich. Außerdem gibt es einige Sprachen welche zwar nicht speziell zur Modellierung von Choreographien entwickelt wurden, sich aber dafür dennoch eignen.

Das Ziel dieser Bachelorarbeit ist zum einen die Durchführung einer Literaturrecherche über diese bereits große Menge an Choreographiesprachen. Es gibt zwar bereits viele Arbeiten,




**(a)** Orchestrierung **(b)** Choreographie

**Abbildung 1.1:** Übersicht von Orchestrierungen und Choreographien [1]

in denen einzelne Sprachen zur Modellierung von Choreographien miteinander verglichen oder vorgestellt wurden, jedoch gibt es bisher nicht viele Arbeiten in denen die mittlerweile große Menge an Choreographiesprachen gesammelt und analysiert wurden. Der erste Teil dieser Bachelorarbeit beschäftigt sich deshalb damit, eine Liste aller existierenden, potentiellen Choreographiesprachen anzufertigen, um einen Überblick über diese große Menge zu bekommen.

Da sich allerdings diese Choreographiesprachen teilweise sehr deutlich voneinander unterscheiden und jeweils unterschiedliche Modellierfähigkeiten bieten, kann man nur mit einer Liste der gesammelten Sprachen noch nicht viel anfangen. Deshalb beschäftigt sich der zweite Teil der Arbeit damit, Kriterien zu erarbeiten, welche von Choreographiesprachen erfüllt werden sollen und basierend auf diesen Kriterien die gesammelten Choreographiesprachen zu analysieren. Da eine Analyse über alle gefunden Sprachen den zeitlichen Rahmen einer Bachelorarbeit sprengen würde, wird nur eine geeignete Teilmenge dieser Choreographiesprachen analysiert. Zur Auswahl einer geeigneten Teilmenge wird im Rahmen dieser Arbeit die Relevanz der jeweiligen Sprachen bestimmt und miteinander verglichen. Ein besonderer Wert wird dabei auf die Datenmodellierungsfähigkeiten der jeweiligen Choreographiesprachen gelegt, da der Austausch von Daten zwischen den einzelnen Teilnehmern einer Choreographie eine wichtige Rolle spielt.

Das Ergebnis dieser Bachelorarbeit ist neben der Sammlung von Choreographiesprachen, auch noch eine Analyse einer Teilmenge dieser Sprachen. Durch die Definition von Kriterien, welche von Choreographiesprachen erfüllt werden sollen, wird in dieser Bachelorarbeit ebenfalls eine Richtlinie definiert, an welche sich zukünftig entwickelte Choreographiesprachen oder Erweiterungen bereits existierender Choreographiesprachen, halten sollten.

[1] http://stackoverflow.com/questions/4127241/orchestration-vs-choreography

## Gliederung

Die Arbeit ist in folgender Weise gegliedert:

**Kapitel 2:** Im Kapitel „Verwandte Arbeiten“ werden existierende Arbeiten vorgestellt, in denen bereits Sprachen zur Modellierung von Choreographien gesammelt und/oder miteinander verglichen wurden.

**Kapitel 3:** Hier wird auf die Suche nach den Sprachen bzw. Modellierungsarten von Choreographien eingegangen. Dabei wird zum einen die Durchführung der Suche erläutert und zum anderen deren Ergebnisse präsentiert. Ebenfalls werden in diesem Kapitel verschiedene Ansätze für die Bestimmung der Relevanz der verschiedenen Sprachen diskutiert. Zum Schluss wird dann nach der Auswahl der Metrik zur Bestimmung der Relevanz, eine Teilmenge an Choreographiesprachen herausgearbeitet, welche im anschließenden Kapitel analysiert werden.

**Kapitel 4:** In diesem Kapitel werden zunächst Vergleichskriterien definiert. Danach werden die ausgewählten Sprachen, aus dem Kapitel zuvor, kurz vorgestellt und anschließend basierend auf diesen Vergleichskriterien analysiert. Die jeweiligen Choreographiesprachen werden anhand von Beispielen veranschaulicht. Den Abschluss des Kapitels bildet eine Übersicht der Ergebnisse der Analyse.

**Kapitel 5:** Den Abschluss der Arbeit liefert eine Zusammenfassung dieser Bachelorarbeit, sowie einen Ausblick darauf, was in zukünftigen Arbeiten beachtet werden soll und wie diese Ergebnisse dieser Bachelorarbeit als Ausgangspunkt für weitere Arbeiten verwendet werden können.

# 2 Verwandte Arbeiten

Es gibt bereits einige Arbeiten, in welchen ausgewählte Sprachen zur Modellierung von Choreographien miteinander verglichen werden. Diese Arbeiten legen ihren Wert dabei alle auf unterschiedliche Kriterien. Der Grund dafür ist auch, dass es keine allgemeingültige Liste von Kriterien für Choreographiesprachen gibt und somit jeder für sich selbst überlegen muss, welche Kriterien man für wichtig empfindet, die eine solche Sprache erfüllen soll.

In der Arbeit von Decker et al. [DKLW09] werden Sprachen auf ihre Fähigkeiten zur Modellierung von Service Choreographien untersucht. Die Kriterien, welche dabei für die Analyse herangezogen werden sind dabei vielfältig. Unter anderem werden hierbei Fragestellungen wie „Was passiert wenn ein Service die Deadline nicht einhält", „Welche Services spielen im Geschäftsprozess überhaupt eine Rolle?" oder „Ist eine Integration mit Orchestrierungssprachen wie BPEL möglich?" als wichtig definiert. Deshalb werden ein paar dieser Kriterien auch für unsere Analyse übernommen. Kriterien zur Datenmodellierungsfähigkeiten werden in dieser Arbeit allerdings nicht herangezogen.

In der Arbeit von Schönberger [Sch11] werden zunächst Choreographien noch einmal in *Service Choreographien*, *B2Bi Choreographien* und *Konzeptionelle Choreographien* unterteilt und diese Kategorien definiert. Im Anschluss werden ebenfalls Choreographiesprachen aus den drei definierten Kategorien miteinander verglichen. Neben der Art der Modelle (Interaktions- vs. Verbindungsmodell) und der Implementierungsunabhängigkeit, spielen auch Kriterien, wie die Ausführbarkeit der Sprache, Möglichkeiten zur Fehlerbehandlung oder die Zerlegbarkeit der Modelle eine Rolle. Kriterien bezüglich der Modellierung von Datenobjekten oder Datenflüssen werden in dieser Arbeit nicht untersucht.

Schönberger, Wilms und Wirtz haben eine Liste von Anforderungen für B2Bi Prozesse erstellt [SWW10]. Es gibt eine Choreographieebene in B2Bi, wodurch sich aus der Liste der Anforderungen eine Teilmenge ableiten lässt, welche man für B2Bi Choreographien anwenden kann. Dafür werden auch die definierten Anforderungen mit der abstrakten Ebene der verschiedenen Modelle, unter anderem Choreographiemodell, verglichen basierend auf Übereinstimmungen.

In der Arbeit von Su et al. [SBFZ08] wird hingegen hauptsächlich Wert auf den Kontrollfluss gelegt. Hierbei ist neben der Modellierung von Aktivitäten und Nachrichten wichtig, ob man die einzelnen Aufgaben parallel oder sequentiell ausführen kann, sowie auch je nach Bedingungen auswählen/entscheiden kann, ob eine Aktivität ausgeführt wird. Ein weiteres Kriterium in dieser Arbeit ist, ob man eine Aufgabe rekursiv aufrufen kann. Wie das Nachrichtenmodell der

einzelnen Sprachen aussieht wird ebenfalls erfasst. Da der Kontrollfluss sehr wichtig ist und es möglich sein sollte, dass verschiedene Aktivitäten erst basierend auf den vorhergehenden Ereignissen ausgeführt oder eben nicht ausgeführt werden, wird in dieser Arbeit das „Choice"-Kriterium, welches den anschließenden Kontrollfluss leitet, übernommen. Im Normalfall ist eine sequentielle Abfolge von Aktivitäten modellierbar, weshalb dieses Kriterium überflüssig erscheint. Es ist uns ebenfalls keine Sprache bekannt, welche zwar „Choice"-Konstrukte zulassen, aber keine zur parallelen Abfolge von Aktivitäten. Da diese Bachelorarbeit allerdings großen Wert auf Datenmodellierungsfähigkeiten legt, verfeinern wir dieses Kriterium, indem wir uns auf Daten-basierte Abfragen begrenzen, wodurch die Wahl („Choice") des Kriteriums automatisch durch bestimmte Daten ermittelt werden. Kriterien bezüglich der Modellierung von Datenobjekten oder Datenflüssen gibt es in dieser Arbeit ebenfalls nicht.

Ein breiteres Spektrum an Kriterien liefert die Arbeit von Bernauer et al. [BKKR03], in welcher Sprachen zur Modellierung von organisationsübergreifenden Workflows verglichen werden. Die Kriterien in dieser Arbeit sind basierend auf unterschiedlichen Perspektiven definiert. Diese sind die „Funktionale Perspektive" (z.B. die Semantik einer Aktivität), „Operationale Perspektive" (wie die Aktivitäten implementiert werden), „Verhaltensperspektive" (z.B. Zeit-Constraints und Fehlerbehandlung), „Informationelle Perspektive" (z.B. wiederverwendbare Datentypen und Datenfluss), „Interaktionsperspektive" (z.B. Interaktions-Primitive), „Organisationsperspektive" (z.B. Rollen) und „Transaktionsperspektive" (z.B. organisationsübergreifende Transaktionen). Diese Arbeit ist jedoch aus dem Jahre 2003, weshalb nur ältere Sprachen untersucht wurden. Im Laufe der Jahre sind allerdings einige neue Choreographiesprachen hinzugekommen. Ein weiteres Problem dieser Arbeit ist, dass zwar eine große Menge an Kriterien für Choreographiesprachen definiert wurden, aber dennoch ist die Analyse der einzelnen Choreographiesprachen in dieser Arbeit lückenhaft. Es wird hier nämlich nicht bei jeder Sprache auf jedes Kriterium eingegangen.

In [MSW11] werden von Meyer, Smirnov und Weske Daten-basierte Kriterien für Modellierungssprachen für Geschäftsprozesse definiert und diese dann auch auf ein paar Sprachen angewendet. Hier werden aber allgemein Sprachen zur Modellierung von Geschäftsprozessen analysiert und nicht speziell Choreographiesprachen.

Eine Literaturrecherche, durchgeführt von Leite et al., findet sich in [LAN+13]. Hier wird eine Literaturrecherche im Bereich Service Choreographie durchgeführt, in denen sie basierend auf verschiedenen Daten Choreographiesprachen sammeln und analysieren. Die gesammelten Sprachen werden ebenfalls basierend auf der Herangehensweise der Sprachen (Modell-basiert, Multi Agenten, etc.) unterteilt. Die Suche verläuft hierbei größtenteils Skript-basiert, wobei im Nachhinein die Menge an Sprachen gefiltert werden, welche bestimmte definierte Kriterien nicht erfüllen. Da nicht manuell noch nach weiteren Sprachen gesucht wird, können Sprachen, welche eventuell für Choreographien geeignet sind, aber nicht speziell dafür entwickelt wurden, teilweise nicht gefunden werden. Daten-bezogene Kriterien werden in dieser Arbeit nicht untersucht.

In der Doktorarbeit von Mancioppi [AM15] werden ebenfalls einige Choreographiesprachen gesammelt und kurz vorgestellt. Eine Analyse bezüglich der Fähigkeit zur Datenmodellierung fehlt hier jedoch.

In den meisten Arbeiten, welche sich mit Choreographiesprachen beschäftigen, wird nur ein geringer Teil an Choreographiesprachen analysiert, wobei sich diese auch oft in den verschiedenen Arbeiten überschneiden. Dabei wird die Analyse oft nur auf bestimmte Kategorien von Choreographiesprachen, wie Sprachen zur Modellierung von Service Choreographien, beschränkt. In einigen Arbeiten werden für die Analyse der Sprachen keine Daten-bezogene Kriterien definiert. Literaturrecherchen bezüglich Choreographiesprachen wurden bisher ebenfalls nicht viele durchgeführt. Die Arbeit von Leite et al. beschränkt sich hierbei auch nur auf Service Choreographiesprachen. Es gibt bisher keine Arbeit, welche eine Literaturrecherche mit einer Analyse kombiniert, wobei die Sprachen auch auf Datenmodellierungsfähigkeiten überprüft werden.

# 3 Identifikation und Sammlung von Choreographiesprachen

In diesem Kapitel geht es um die Suche nach den Choreographiesprachen. Zunächst einmal wird die Herangehensweise der Suche erläutert. Daraufhin wird das Ergebnis der Suche präsentiert. Zum Abschluss wird dann die Relevanz der einzelnen Sprachen ermittelt, auf Basis derer eine ausgewählte Anzahl an Sprachen für die weitere Analyse identifiziert wird.

## 3.1 Herangehensweise der Suche

Die Grundlage der Suche ist eine von Schönberger et al. erarbeitete Liste [SHKW11], in welcher bereits einige mögliche Sprachen zur Modellierung von Choreographien gesammelt werden. Der Großteil der Sprachen wird hierbei Skript-basiert gesucht. Einige dieser Sprachen dienen jedoch nicht zur Modellierung von Choreographien, weshalb die Menge der Sprachen in dieser Bachelorarbeit manuell überprüft wird, ob diese Sprachen wirklich zur Modellierung von Choreographien geeignet sind. Die Sprachen, welche in anderen Artikeln oder direkt im Definitionspaper jedoch als Choreographiesprachen definiert werden, werden in die Liste aufgenommen. Zusätzlich werden noch weitere Sprachen in die Liste aufgenommen, welche zur Modellierung von Choreographien geeignet sind, basierend auf der Definition der jeweiligen Sprache.

Es gibt drei verschiedene Wege nach weiteren Choreographiesprachen zu suchen, welche hier angewendet werden:

- Rekursiv durch die Referenzen der Paper der bereits gefundenen Sprachen
- Ebenso durch eine Menge von Artikel, die eben diese Paper selbst referenzieren
- Durch *Google Scholar*[1] mithilfe der Namen der verschiedenen untergeordneten Choreographiearten: „Service Choreography“, „Process Choreography“, „B2Bi Choreography“, sowie „Conceptual Choreography“.

[1] https://scholar.google.de/

Besonders wird dabei auf kürzlich publizierte Artikel geachtet, da die ursprüngliche Liste [SHKW11] bereits im Jahr 2011 erstellt wurde und somit bereits eine große Menge der Sprachen beinhaltet, welche bereits vorher erschienen sind.

## 3.2 Ergebnisse der Suche

Tabelle 3.1 zeigt das Ergebnis der Suche nach Choreographiesprachen. Diese Tabelle enthält alle gefundenen, potentiellen Choreographiesprachen, jedoch wird die Anzahl der Spalten hier auf die relevanteste Informationen beschränkt. Die ausführlichere angefertigte Tabelle enthält noch ein paar Informationen mehr, wie z.B. der Name des Artikels oder von welchem Verlag der Artikel veröffentlicht wurde. In Tabelle 3.1 wird neben dem Namen der Sprache auch noch die Anzahl der Zitationen der Publikation in der die Sprache definiert wird und der Relevanzwert der Publikation angegeben. Die Referenz von dem jeweiligen Artikel ist hier auch angegeben. Wie der Relevanzwert errechnet wird, wird in Abschnitt 3.4 erläutert. Zusätzlich dazu wird noch das Jahr angegeben, in welchem das Paper erschienen ist bzw. die Sprache veröffentlicht wurde.

| **Name der Sprache** | **Jahr** | **#Zitationen** | **Relevanzwert** | **Referenz** |
|---|---|---|---|---|
| Amoeba | 2009 | 87 | 10,88 | [DCS09] |
| Abstract CDL (ACDL) | 2009 | 27 | 3,38 | [MBLN09] |
| Artifact-Centric Collaboration Model (ACC) | 2011 | 18 | 3,00 | [YLZ11] |
| BioFlow | 2008 | 27 | 3,00 | [JIH10] |
| Blindingly Simple Protocol Language (BSPL) | 2011 | 31 | 5,17 | [Sin11] |
| „Bologna“[2] | 2005 | 175 | 14,58 | [BGG+05b] |
| BPEL4Chor | 2007 | 299 | 29,90 | [DKLW07] |
| BPEL$^{gold}$ | 2010 | 15 | 2,14 | [KEL+11] |
| BPEL$^{light}$ | 2007 | 63 | 6,30 | [NLKL07] |
| Business Process Modeling Notation 1.0 (BPMN 1.0) | 2004 | 753 | 57,92 | [Whi04] |
| BPMN 2.0 Choreographiediagramm | 2011 | 171 | 28,50 | [Mod11] |
| BPMN 2.0 Kollaborationsdiagramm | 2011 | 171 | 28,50 | [Mod11] |
| BPMN 2.0 Konversationsdiagramm | 2011 | 171 | 28,50 | [Mod11] |
| Business Choreography Language (BCL) | 2009 | 12 | 1,50 | [MZW09] |

[2]Die Sprache wurde im Artikel nicht benannt. In [SBFZ08] wurde sie unter dem Namen „Bologna“ referenziert, da sie an der Universität von Bologna entwickelt wurde.

| Name der Sprache | Jahr | #Zitationen | Relevanzwert | Referenz |
|---|---|---|---|---|
| Business Process Definition Metamodel (BPDM) | 2003 | 3 | 0,21 | [Koe03] |
| Business Transaction Net (BTx-Net) | 2007 | 13 | 1,30 | [SY07] |
| CDL | 2006 | 84 | 7,64 | [YZQ+06] |
| $CDL_0$ | 2006 | 21 | 1,91 | [LHPZ06] |
| Chor | 2007 | 154 | 15,40 | [QZCY07] |
| $Chor_c$ | 2008 | 12 | 1,33 | [QCP+08] |
| Choreographical Business Transaction Net (CoBTx-Net) | 2007 | 4 | 0,40 | [SY08] |
| Choreography Calculus (CC) | 2014 | 25 | 8,33 | [Mon14] |
| Choreography Language (CL) | 2005 | 49 | 4,08 | [BGG+05a] |
| CHOReOS | 2010 | 13 | 1,86 | [VIG+10] |
| ChorTex | 2015 | 0 | 0,00 | [AM15] |
| Circulate | 2009 | 23 | 2,88 | [BWH09] |
| Colaba | 2011 | 7 | 1,17 | [CS11] |
| Collaborative Structure of MAS for the Web Services Composition (CS-MWC) | 2007 | 6 | 0,60 | [XQH+07] |
| Colombo | 2005 | 330 | 27,50 | [BCD+05] |
| Colored Petri Nets (CPN) | 1981 | 786 | 21,83 | [Jen81] |
| Communicating Sequential Processes (CSP) | 2006 | 40 | 3,64 | [Yeu06] |
| Community Process Definition Language (CPDL) | 2002 | 7 | 0,47 | [SHK02] |
| Composition And Semantic Enhancement of Web Services (CASheW-S) | 2005 | 37 | 3,08 | [NFH05] |
| Compositional Choreographies | 2013 | 43 | 10,75 | [MY13] |
| Conditional Pi-calculus (Cpi-calculus) | 2009 | 7 | 0,88 | [ZZ09] |
| Contextual Aspect-Sensitive Services (CASS) | 2005 | 24 | 2,00 | [CE05a] |
| Contract Definition Language (CDL) | 2006 | 26 | 2,36 | [Mil06] |
| Contract Workflow Model (CWM) | 2003 | 1 | 0,07 | [Kab03] |
| Cooperative Agentflow Process Language (CA-PLAN) | 2003 | 21 | 1,50 | [YW03] |
| CoopFlow | 2005 | 12 | 1,00 | [CT05] |

| Name der Sprache | Jahr | #Zitationen | Relevanzwert | Referenz |
|---|---|---|---|---|
| Data-Aware Choreography (DAChor) | 2012 | 25 | 5,00 | [KPR12] |
| „Declarative Choreography Language“[3] | 2012 | 15 | 3,00 | [SXS12] |
| DecSerFlow | 2006 | 394 | 35,82 | [AP06a] |
| Delta Distributed States Temporal Logic (DeltaDSTL) | 2006 | 13 | 1,18 | [MS06] |
| Deterministic Finite State Automata (DFA) | 2004 | 193 | 14,85 | [WFMN04] |
| Distributed States Logic (DSL) | 2003 | 4 | 0,29 | [MS03] |
| Distributed States Temporal Logic (DSTL) | 2003 | 4 | 0,29 | [MS03] |
| Distributed Workflow Definition Language (DWDL) | 2010 | 1 | 0,14 | [Ker10] |
| Documentary Petri Nets (DPN) | 1995 | 87 | 3,95 | [BLWW95] |
| Dynamic Choreographies (DYCHOR) | 2006 | 86 | 7,82 | [RWR06] |
| Dynamic Interaction-Oriented Choreography (DIOC) | 2015 | 15 | 7,50 | [DGG+15] |
| Dynamic Process-Oriented Choreography (DPOC) | 2015 | 15 | 7,50 | [DGG+15] |
| Dynamic Workflow Model (DWM) | 2002 | 120 | 8,00 | [MSLH02] |
| ebXML Business Process Specification Schema (ebXML BPSS) | 2001 | 35 | 2,19 | [CCK+01] |
| ebBP-Reg | 2010 | 6 | 0,86 | [SW10] |
| Entish | 2002 | 0 | 0,00 | [Amb02] |
| EPC Markup Language (EPML) | 2004 | 39 | 3,00 | [MN04] |
| eSourcing Markup Language (eSML) | 2011 | 0 | 0,00 | [Nor11] |
| Event-driven Process Chains (EPC) | 1992 | 1151 | 46,04 | [KSN92] |
| Executable Choreography Framework (ECF) | 2005 | 7 | 0,58 | [CE05b] |
| Extended EPCs (eEPC) | 2005 | 29 | 2,42 | [SV05] |
| Federated Choreographies | 2006 | 21 | 1,91 | [ELT06] |

[3] Es wurde kein expliziter Name für die Choreographiesprache in dem Artikel angegeben.

| Name der Sprache | Jahr | #Zitationen | Relevanzwert | Referenz |
|---|---|---|---|---|
| Framework for Orchestration, Composition and Aggregation of Services (FOCAS) | 2009 | 28 | 3,50 | [PE09] |
| Framework for Service Choreography (FSC) | 2015 | 4 | 2,00 | [BRM15] |
| General-Stochastic-Petri-Nets (GSPN) | 2010 | 1 | 0,14 | [XOWY10] |
| Global Calculus | 2007 | 304 | 30,40 | [CHY07] |
| Grid Services Flow Language (GSFL) | 2002 | 276 | 18,40 | [KWV+02] |
| Grid Workflow Execution Language (GWEL) | 2006 | 72 | 6,55 | [Cyb06] |
| Grid-Flow Description Language (GFDL) | 2006 | 97 | 8,82 | [GHB+06] |
| High-level Message Sequence Charts (HMSC) | 1997 | 137 | 6,85 | [MR97] |
| iBPMN | 2007 | 91 | 9,10 | [DB08] |
| Interaction Petri Nets (IPN) | 2007 | 74 | 7,40 | [DW07] |
| Interactive Behavior Model (IABM) | 2009 | 7 | 0,88 | [ZZ09] |
| Interorganizational Workflow Net (IOWF-Net) | 2001 | 376 | 23,50 | [AW01] |
| JavaABC (jABC) | 2006 | 22 | 2,00 | [KNMS06] |
| L3 | 2005 | 0 | 0,00 | [GHB05] |
| „LCC Extension 1“[4] | 2009 | 1 | 0,13 | [BB09] |
| „LCC Extension 2“[5] | 2009 | 1 | 0,13 | [BB09] |
| Let's Dance | 2006 | 201 | 18,27 | [ZBDH06] |
| Lightweight Coordination Calculus (LCC) | 2004 | 112 | 8,62 | [Rob04] |
| LTS-Based Choreography Model (C-LTS) | 2009 | 45 | 5,63 | [MCT09] |
| m3pl | 2006 | 8 | 0,73 | [HO06] |
| MEMO Organisation Modelling Language (MEMO-OrgML) | 2001 | 4 | 0,25 | [FJ01] |
| MEMO Resource Modelling Language (MEMO-ResML) | 2006 | 5 | 0,45 | [Jun06] |
| Message Choreography Model (MCM) | 2009 | 15 | 1,88 | [WRS+09] |

[4]Es wurde eine Erweiterung von LCC angegeben, welche nicht benannt wurde.
[5]Es wurde eine Erweiterung von LCC angegeben, welche nicht benannt wurde.

| Name der Sprache | Jahr | #Zitationen | Relevanzwert | Referenz |
|---|---|---|---|---|
| Message Sequence Charts (MSC) | 1996 | 215 | 10,24 | [RGG96] |
| Multiagent Protocols (MAP) | 2009 | 98 | 12,25 | [BWR09] |
| Multi Metamodel Process Ontology (m3po) | 2006 | 30 | 2,73 | [HOK06] |
| Network Coordination Policies Calculus (NCP Calculus) | 2010 | 11 | 1,57 | [CFGS10] |
| Open Workflow Nets (oWFN) | 2005 | 210 | 17,50 | [MRS05] |
| OWL-P | 2004 | 15 | 1,15 | [DMCS04] |
| p-LTL | 2016 | 0 | 0,00 | [RS16] |
| Pi-Exchange-Model (piX-model) | 2007 | 7 | 0,70 | [STDD07] |
| Prioritized-timed Colored Petri Nets (PTCPN) | 2012 | 12 | 2,40 | [VMP+12] |
| Procedure Constraint Grammar (PCG) | 1997 | 12 | 0,60 | [Lee97] |
| Process Algebra for WS-CDL (PA4WS) | 2008 | 0 | 0,00 | [LM08] |
| q-LTL | 2016 | 0 | 0,00 | [RS16] |
| Q4BPMN | 2012 | 6 | 1,20 | [BBC+12] |
| rBPMN | 2009 | 5 | 0,63 | [MGWH09] |
| Reo | 2009 | 55 | 6,88 | [AKM08] |
| ScriptOrc | 2008 | 8 | 0,89 | [BS08] |
| SENSORIA Reference Modelling Language (SRML) | 2006 | 111 | 10,09 | [FLB06] |
| „Service Choreography“[6] | 2012 | 1 | 0,2 | [SZZ+12] |
| ServiceNet | 2012 | 0 | 0,00 | [LZD12] |
| SOAP Service Description Language (SSDL) | 2005 | 18 | 1,50 | [PWW+05] |
| SOPHIE | 2005 | 2 | 0,17 | [AL05] |
| Subject-Oriented Business Process Management (S-BPM) | 2009 | 72 | 9,00 | [Fle10] |
| TDWF | 2006 | 3 | 0,27 | [SZS06] |
| UML Aktivitätsdiagramm | 2005 | 6945 | 578,75 | [RBJ05] |
| UML Interaktionsübersichtsdiagramm | 2005 | 6945 | 578,75 | [RBJ05] |
| UML Kommunikationsdiagramm | 2005 | 6945 | 578,75 | [RBJ05] |

[6] Hier wird ein formales Modell, basierend auf oWFN's, beschrieben. Dieses wird in dem Artikel als Service Choreography benannt.

| Name der Sprache | Jahr | #Zitationen | Relevanzwert | Referenz |
|---|---|---|---|---|
| UML Sequenzdiagramm | 2005 | 6945 | 578,75 | [RBJ05] |
| UML Zeitdiagramm | 2005 | 6945 | 578,75 | [RBJ05] |
| UML Profile for Collaborative Business Processes based on Interaction Protocols (UP-ColBPIP) | 2004 | 4 | 0,31 | [VSC04] |
| UN/CEFACT'S Modeling Methodology (UMM) | 2006 | 37 | 3,36 | [HHL+06] |
| Web Service Choreography Interface (WSCI) | 2002 | 449 | 29,93 | [AAF+02] |
| Web Services Conversation Language (WSCL) | 2002 | 187 | 12,47 | [BBB+02] |
| Web Services Flow Language (WSFL) | 2001 | 988 | 61,75 | [Ley+01] |
| Web Services Integration and Processing Language (WSIPL) | 2003 | 11 | 0,79 | [CLNL03] |
| Workflow View Definition Language (WfVDL) | 2010 | 1 | 0,14 | [Ker10] |
| Web Services Choreography Description Language (WS-CDL) | 2004 | 598 | 46,00 | [BDO05] |
| WS-CDL+ | 2007 | 25 | 2,50 | [KWH07] |
| Web Service Modeling Ontology (WSMO) Choreography | 2006 | 57 | 5,18 | [RSF+06] |
| Web Service Query Model | 2010 | 24 | 3,43 | [YRB+10] |
| Web Service Transition Language (WSTL) | 2004 | 84 | 6,46 | [BDDM04] |
| XML Process Definition Language (XPDL) | 2002 | 65 | 4,33 | [Int02] |
| XML-Nets | 2001 | 73 | 4,56 | [Len01] |
| XScufl | 2004 | 8 | 0,62 | [Oin04] |
| Yet Another Event-Driven Process Chain (yEPC) | 2005 | 38 | 3,17 | [MNN05] |

**Tabelle 3.1:** Gefundene Choreographiesprachen

**Hinweis:** Bei manchen Sprachen gibt es verschiedene Versionen der Sprachen (wie in BPMN oder UML), deshalb ist es schwer dort einen Artikel als Referenz auszuwählen. Dies gilt ebenfalls für Sprachen bei denen es zwar eine Version gibt, welche aber dennoch in unterschiedlichen Artikeln vorgestellt werden. Im Zweifelsfall wurde der Artikel ausgewählt, welcher die meisten Zitationen hat, jedoch gibt es eventuell auch einen Artikel bei denen ebenfalls eine Sprache

aus der Liste nochmals vorgestellt wird und welcher noch mehr Zitationen hat. Dieser wurde dann lediglich nicht gefunden.

Obwohl BPMN 1.0 bereits durch BPMN 2.0 abgelöst wurde, wird es in der Tabelle aufgelistet. Der Grund ist, dass BPMN 1.0 eine der älteren bekannten Sprachen zur Modellierung von Choreographien ist. Es gibt Choreographiesprachen oder Erweiterungen wie iBPMN [DB08], die direkt auf BPMN 1.0 aufbauen.

Domänen-spezifische Choreographiesprachen wie „RosettaNet“ [7], „Society for Worldwide Interbank Financial Transfer“[8] (kurz: SWIFTNet) oder Health Level Seven[9] (kurz: HL7) wurden nicht in die Tabelle aufgenommen.

Hierbei muss man auch beachten, dass aufgrund der unterschiedlichen Definitionen von Choreographien viele dieser Sprachen eventuell auch eher als Orchestrierungssprachen geeignet sind. Es wurde bei der Suche nach den Sprachen jedoch darauf geachtet, ob diese entweder direkt als Choreographiesprachen vorgestellt werden oder deren Beschreibung zur Grundidee einer Choreographiesprache passt. Sprachen wurden ebenfalls in die Tabelle aufgenommen, wenn sie in anderen Publikationen als Choreographiesprachen angesehen werden. Der Standard für Orchestrierungssprachen „Business Process Execution Language“ (kurz: BPEL; [ACD+03]) wird auch in mehreren Artikeln genannt, um damit Choreographien zu modellieren, wobei die Sprache selbst dafür nicht so gut geeignet ist. Deshalb wurde BPEL nicht in die Tabelle aufgenommen, obwohl es in einigen Artikeln zur Modellierung von Choreographien verwendet wird. Das gleiche Problem gibt es mit „XLANG“ [Tha01], einem der beiden Vorgänger von BPEL, welches wir deshalb ebenfalls nicht in die Tabelle aufgenommen haben.

## 3.3 Möglichkeiten zur Ermittlung der Relevanz der gefundenen Sprachen

Wir haben nun 130 mögliche Sprachen zur Modellierung von Choreographien gefunden. Wenn man die Modellierungssprachen nun miteinander vergleichen will, dann muss diese Menge an Sprachen für die weitere Analyse gefiltert werden, da die Menge zu groß ist, um eine Analyse über alle gefundenen möglichen Choreographiesprachen in einer Bachelorarbeit durchzuführen. Nun stellt sich aber die Frage, welche Sprachen für die weitere Analyse ausgewählt werden sollen?

**Idee 1:** Die erste Idee ist, dass man eine Formel aufstellt, um die Relevanz der Sprachen zu ermitteln. Eine entsprechende Formel zur Berechnung der Relevanz einer Sprache kann beispielsweise wie folgt aussehen:

[7] http://resources.gs1us.org/
[8] https://www.swift.com/
[9] http://www.hl7.org/

$$Rel(ML) = \alpha \cdot c(mp) + \beta \cdot \sum_{s_i \in S_{all}} Imp(p|s_i) + \gamma \cdot \sum_{s_j \in S_{new}} Imp(p|s_j)$$

Diese beinhaltet: (1) Die Anzahl der Zitationen des Artikels in dem die Sprache definiert wurde ($c(mp)$), (2) Die Anzahl der Treffer, wenn man nach dem Akronym der Sprache in *Google Scholar* sucht zusammen mit dem Begriff „Choreography“ und unterschiedlichen Parametern, wie (2.1) zeitliche Einschränkungen und (2.2) Publisher Rating. $Imp(p|s_i)$ soll hier die Wichtigkeit des gefundenen Artikels $s_i$ spiegeln, basierend auf dem Publisher $p$. Der Grund hierfür ist, dass renommierte Verlage, wie Springer, IEEE, ACM oder Elsevier im Normalfall hochwertige Artikel veröffentlichen, da die Artikel und deren Qualität vor der Veröffentlichung von Experten begutachtet werden. Über andere Kanäle veröffentlichte Artikel sind teilweise von der Qualität her nicht immer hochwertig, da im Endeffekt jeder etwas auf irgendeiner Plattform veröffentlichen kann. Deshalb ist die Idee, dass die Artikel basierend auf dem Publisher gewichtet werden. Die Summe $\sum_{s_i \in S_{all}}$ steht dafür, dass alle gefundenen Treffer der Suchanfrage zur Berechnung der Relevanz miteinbezogen werden. $\sum_{s_j \in S_{new}}$ steht dabei für die Anzahl der Treffer der letzten $x$ Jahre. Die Treffer der letzten Jahre sollen stärker gewichtet werden, da neuere Sprachen in der Regel noch weniger zitiert sind, als bereits etablierte Sprachen. Da bei den Treffern in Google auch Artikel angezeigt werden, in denen die gesuchte Sprache nur kurz erwähnt wird, macht es für die wichtigen Sprachen keinen Unterschied wann sie definiert wurden, da eben auch die wichtigsten Sprachen von früher, welche die Grundbausteine legen, im Normalfall immer noch in neuen Artikeln erwähnt werden. Ein weiterer Punkt die neuen Treffer höher zu gewichten, ist die Tatsache dass so kürzlich entwickelte Sprachen eine größere Chance haben, analysiert zu werden. Wenn eine Sprache schon lange existiert, dann wird sie häufig auch öfters erwähnt als neuere Sprachen. Durch die Gewichtung der Zeit in der die jeweiligen Artikel erschienen sind, in der eine solche Sprache definiert wird, wird diesem Effekt entgegengewirkt. Wie genau $S_{new}$ definiert, müsste man näher diskutieren, insofern man sich für diesen Ansatz zur Berechnung der Relevanz einer Sprache entscheidet.

**Problem:** Die gleichen Abkürzungen werden teilweise mehrfach verwendet (z.B. CDL) oder sie sind in anderen Wörtern enthalten (z.B. MAP in map, mapping, etc.), weshalb dies das Ergebnis verfälschen würde. Man kann jedoch anstatt den Akronymen der Choreographiesprachen die ausgeschriebene Form zur Suche verwenden. Das Problem dabei ist dass für viele Sprachen, vor allem die geläufigeren Choreographiesprachen, in einigen Artikeln nur das Akronym der Sprache erwähnt wird. Ein weiteres Problem ist, dass viele Artikel, welche den Suchbegriff „Choreography“ beinhalten, sich sehr oft nicht direkt um solche Sprachen drehen. Da WS-CDL eine sehr bekannte Sprache ist und auch das Wort „Choreography“ beinhaltet, wird diese auch oft in Artikeln verwendet in denen es nicht direkt um Choreographiesprachen geht. Dies würde also das Ergebnis ebenfalls beeinträchtigen.

**Idee 2:** Eine zweite Lösung ist, dass man nur die Anzahl der Zitationen von dem Artikel berücksichtigt, in welchem die jeweilige Sprache definiert wird, also:

$$Rel(ML) = c(mp)$$

**Problem:** Ein generelles Problem hier ist, dass viele Artikel, in welchen diese Arbeiten referenziert werden, nicht direkt etwas mit Choreographiesprachen zu tun haben. Hier werden also nicht nur Artikel gezählt, in denen es um Choreographiesprachen geht, sondern alle Artikel werden berücksichtigt, die die jeweilige Publikation referenziert haben. Bei manchen Sprachen gibt es auch verschiedene Versionen der Sprachen (wie in BPMN oder UML), deshalb ist es schwer dort ein Artikel als Definitionspaper auszuwählen. Dies gilt ebenfalls für Sprachen bei denen es zwar nur eine Version gibt, welche aber dennoch in unterschiedlichen Arbeiten vorgestellt werden. Ein zusätzliches Problem besteht darin, dass neuere Choreographiesprachen im Gesamten weniger zitiert werden, da sie eben nur in noch neueren Artikel zitiert werden können.

**Idee 3:** Die dritte Idee ist, dass man die Anzahl der Zitationen vom Hauptartikel gewichtet, sodass das letzte Problem der zweiten Idee wegfällt. Als Nenner kann man hier dann die Anzahl der Jahre nehmen, die vergangen sind seit der ersten Publikation der jeweiligen Sprache, also:

$$Rel(ML) = \frac{c(mp)}{2017 - Erscheinungsjahr}$$

**Problem:** Hier fällt das letzte Problem aus der zweiten Idee weg. Die anderen zwei Probleme aus der zweiten Idee bleiben jedoch bestehen. Ein Problem, welches hier hinzukommen kann, ist dass wenn eine Sprache erst dieses Jahr veröffentlicht worden ist, die Berechnung einen Fehler liefern würde, wegen einer $0$ im Nenner. Hier muss man dann manuell als Nenner die Zahl $1$ nehmen. Allerdings haben wir bei der Suche nach Choreographiesprachen, keine Sprachen gefunden, welche erst in diesem Jahr veröffentlicht wurden, weshalb dieser Fehler in unserem Fall nie auftritt.

## 3.4 Berechnung der Relevanz zur Filterung der Sprachen für die Analyse

Es bleibt zu erwähnen, dass es keine optimale Lösung zur Auswahl einer geeigneten Teilmenge von Choreographiesprachen für die weitere Analyse gibt, jedoch erscheint die dritte Idee in unserem Fall am besten zu sein. Wir wählen alle Sprachen mit einem Relevanzwert $\geq 12$ für die folgende Analyse aus. In Tabelle 3.2 sind die 27 übrig gebliebenen möglichen Choreographiesprachen, sortiert nach ihrem Relevanzwert, aufgelistet.

| Name der Sprache | Jahr | #Zitationen | Relevanzwert | Referenz |
|---|---|---|---|---|
| UML Aktivitätsdiagramm | 2005 | 6945 | 578,75 | [RBJ05] |
| UML Interaktionsübersichtsdiagramm | 2005 | 6945 | 578,75 | [RBJ05] |
| UML Kommunikationsdiagramm | 2005 | 6945 | 578,75 | [RBJ05] |
| UML Sequenzdiagramm | 2005 | 6945 | 578,75 | [RBJ05] |
| UML Zeitdiagramm | 2005 | 6945 | 578,75 | [RBJ05] |
| Web Services Flow Language (WSFL) | 2001 | 988 | 61,75 | [Ley+01] |
| Business Process Modeling Notation 1.0 (BPMN 1.0) | 2004 | 753 | 57,92 | [Whi04] |
| Event-driven Process Chains (EPC) | 1992 | 1151 | 46,04 | [KSN92] |
| Web Services Choreography Description Language (WS-CDL) | 2004 | 598 | 46 | [BDO05] |
| DecSerFlow | 2006 | 394 | 35,82 | [AP06a] |
| Global Calculus | 2007 | 304 | 30,4 | [CHY07] |
| Web Service Choreography Interface (WSCI) | 2002 | 449 | 29,93 | [AAF+02] |
| BPEL4Chor | 2007 | 299 | 29,9 | [DKLW07] |
| BPMN 2.0 Choreographiediagramm | 2011 | 171 | 28,5 | [Mod11] |
| BPMN 2.0 Kollaborationsdiagramm | 2011 | 171 | 28,5 | [Mod11] |
| BPMN 2.0 Konversationsdiagramm | 2011 | 171 | 28,5 | [Mod11] |
| Colombo | 2005 | 330 | 27,5 | [BCD+05] |
| Interorganizational Workflow Net (IOWF-Net) | 2001 | 376 | 23,5 | [AW01] |
| Colored Petri Nets (CPN) | 1981 | 786 | 21,83 | [Jen81] |
| Grid Services Flow Language (GSFL) | 2002 | 276 | 18,4 | [KWV+02] |
| Let's Dance | 2006 | 201 | 18,27 | [ZBDH06] |
| Open Workflow Nets (oWFN) | 2005 | 210 | 17,5 | [MRS05] |
| Chor | 2007 | 154 | 15,4 | [QZCY07] |

| Name der Sprache | Jahr | #Zitationen | Relevanzwert | Referenz |
|---|---|---|---|---|
| Deterministic Finite State Automata (DFA) | 2004 | 193 | 14,85 | [WFMN04] |
| „Bologna“[10] | 2005 | 175 | 14,58 | [BGG+05b] |
| Web Services Conversation Language (WSCL) | 2002 | 187 | 12,47 | [BBB+02] |
| Multiagent Protocols (MAP) | 2009 | 98 | 12,25 | [BWR09] |

**Tabelle 3.2:** Sprachen nach der Filterung

Dies sind nun die Sprachen, welche im nächsten Kapitel genauer analysiert werden, mit drei Ausnahmen: WSCI [AAF+02] und WSCL [BBB+02] sind beides Vorgänger der Sprache WS-CDL [BDO05], welches eine Kombination aus beiden Sprachen mit zusätzlichen Eigenschaften ist. Deshalb wird nur die neuere Sprache WS-CDL analysiert. Das gleiche gilt für BPMN 1.0, welches ebenfalls nicht betrachtet wird, da es nur eine ältere Version von BPMN 2.0 Kollaborationsdiagrammen ist. Also werden die restlichen 24 potentiellen Choreographiesprachen im nächsten Kapitel untersucht.

[10] Die Sprache wurde im Artikel nicht benannt. In [SBFZ08] wurde sie unter dem Namen „Bologna“ referenziert, da sie an der Universität von Bologna entwickelt wurde.

# 4 Analyse ausgewählter Sprachen

In diesem Kapitel werden zunächst die Kriterien für die Analyse vorgestellt. Anschließend werden die ausgewählten Choreographiesprachen kurz vorgestellt und basierend auf diesen Kriterien analysiert. Die Analyse wird dabei mithilfe von Beispielmodellierungen unterstützt.

## 4.1 Vergleichskriterien

Bevor man Choreographiesprachen analysieren und miteinander vergleichen kann, benötigt man vordefinierte Kriterien, in welchen Punkten man sie miteinander vergleicht. In Kapitel 2 wurden bereits verwandte Arbeiten kurz vorgestellt. Dabei wurden ebenfalls ein paar Kriterien für Choreographiesprachen erwähnt, welche in den jeweiligen Arbeiten definiert werden. Da wir in dieser Bachelorarbeit eine große Mengen an Sprachen analysieren, beschränken wir uns in der Analyse nur auf ein paar grundlegende Kriterien, wobei speziell Daten-bezogene Kriterien herangezogen werden. Im Folgenden werden die Kriterien definiert, welche wir für die Analyse verwenden. Die ersten drei Kriterien werden aus [DKLW09], die drei darauf folgenden Kriterien aus [MSW11] übernommen. Manche dieser Kriterien werden auch in anderen Arbeiten verwendet, welche in Kapitel 2 („Verwandte Arbeiten“) vorgestellt wurden. Das letzte Kriterium wird in [HBKL17] angesprochen.

- *(K1) Multilaterale Interaktionen:* In den meisten unternehmensübergreifenden Geschäftsprozessen sind mehr als nur ein Teilnehmer involviert. Es muss deshalb möglich sein, dass mehrere Teilnehmer in einer Choreographie miteinander kommunizieren können. Deshalb ist das erste Kriterium, dass eine Sprache multilaterale Interaktionen unterstützen muss.
- *(K2) Teilnehmertopologie:* Da in den meisten unternehmensübergreifenden Geschäftsprozessen mehrere Partner involviert sind, sollten Choreographiesprachen ein globale Sicht auf alle involvierten Teilnehmer unterstützen. Dies erlaubt einen besseren Überblick der Teilnehmer einer Choreographie und deren Interaktionen untereinander. Solch eine Übersicht wird Teilnehmertopologie genannt.
- *(K3) Zeit-Constraints:* In jedem unternehmensübergreifenden Geschäftsprozess spielt die Zeit eine ganz zentrale Rolle. Es gibt oft Fristen bis wann eine Aufgabe erfüllt sein muss oder wie lange man auf eine Antwort von einem anderen Teilnehmer wartet. Dies ist beispielsweise notwendig wenn ein Teilnehmer eine Anfrage an mehrere Partner sendet

und dann auf deren Antworten wartet. Hier helfen Zeit-Constraints (Constraints sind Beschränkungen) den Ablauf einer Choreographie zu garantieren, falls einer oder mehrere Partner gar nicht oder nicht innerhalb eines bestimmten Zeitfensters auf die Anfrage antworten. Deshalb sollten Choreographiesprachen die Modellierung von Zeit-Constraints unterstützen. Ein Beispiel wäre hierbei: Teilnehmer 1 braucht wichtige Dokumente bis zur Frist $t$. Diese können nur von Teilnehmer 2 geliefert werden. Teilnehmer 1 sendet also eine Anfrage an Teilnehmer 2. Daraufhin wartet Teilnehmer 1 auf eine Antwort, welche die entsprechenden angefragten Dokumente enthält bis zur Frist $t$. Falls er sie bis dahin noch nicht erhalten hat, wird er einen alternativen Pfad gehen.

- *(K4) Daten-basierter Kontrollfluss:* In vielen Arbeitsprozessen müssen oft Entscheidungen getroffen werden für das weitere Vorgehen, d.h. welcher Pfad mit welchen Aufgaben als nächstes ausgeführt werden soll. Diese sind oft von bestimmten Daten abhängig. Deshalb ist ein weiteres wichtiges Kriterium, dass man einen Kontrollfluss Daten-basiert modellieren kann, indem man einzelne Daten abfragt. Das können Abfragen sein, ob der Wert eines Datenobjekts größer, kleiner oder gleich der Wert eines anderen Datenobjekts oder als ein bestimmter Grenzwert ist. Außerdem lassen sich komplexe logische Ausdrücke über ein oder mehrere Datenobjekte abfragen. Basierend auf dem Ergebnis der Evaluierung dieser Abfrage wird der entsprechende Pfad gewählt, welcher die Bedingungen erfüllt.
- *(K5) Modellierung von Datenobjekten:* In Choreographien werden durch Interaktionen Daten ausgetauscht. Deshalb sollten die Sprachen eine explizite Modellierung von Datenobjekten unterstützen, damit sich relevante Choreographiedaten modellieren lassen, welche untereinander ausgetauscht werden können. Somit hat man eine globale Sicht auf diese relevante Daten. Ebenfalls lassen sich dadurch wichtige Daten modellieren, welche zwar nicht mit anderen Teilnehmern ausgetauscht werden, welche aber relevant für einzelne Aktivitäten eines bestimmten Teilnehmers sind.
- *(K6) Expliziter Datenfluss:* Neben der Modellierung von Datenobjekten, sollte es auch noch einen expliziten Datenfluss geben, mit welchem man modellieren kann, welche Aufgaben bestimmte Daten verwenden (Input einer Aufgabe) oder welche Daten durch diese Aufgaben erzeugt werden (Output einer Aufgabe).
- *(K7) Partnerübergreifender Datenfluss:* In Choreographiesprachen finden die Interaktionen oft nur durch Nachrichtenaustausch statt, weshalb auch Daten nur per Nachrichten an andere Teilnehmer gesendet werden können. Die Möglichkeit zur Modellierung von explizitem Datenfluss zwischen Teilnehmern einer Choreographie ohne deren Einbettung in Nachrichten reduziert den Modellierungsaufwand und die Komplexität der Modelle [HBKL17]. Deshalb ist ein weiteres Kriterium, dass Daten zwischen den Teilnehmern ausgetauscht werden können, ohne dass man diese Daten in Nachrichten einbetten und verschicken muss. Dieses Kriterium nennen wir „Partnerübergreifender Datenfluss“.

## 4.2 Abstraktes Beispiel einer Choreographie

Da eine Analyse der Sprachen mit einem dazugehörigen Beispiel besser veranschaulicht werden können, wird hier zunächst ein abstraktes Beispiel eingeführt, welches alle Kriterien visualisiert. Bei der Analyse der einzelnen Choreographiesprachen wird sofern möglich dieses Beispiel mit der jeweiligen Sprache umgesetzt. Da eben nicht mit jeder Choreographiesprache jedes Kriterium modelliert werden kann, lassen sich mit einigen Sprachen nur Teile des Beispiels umsetzen. Außerdem muss beachtet werden, dass es nicht für jede Choreographiesprache kostenlose und öffentlich zugängliche Werkzeuge zur Modellierung gibt, weshalb dieses abstrakte Beispiel nicht für jede Choreographiesprache modelliert werden kann. In dem Fall, dass es keine geeigneten Werkzeuge zur Modellierung einer Choreographiesprache gibt, wird dann ein Beispiel aus der Quelle der Choreographiesprache verwendet.

**Beispiel:** Im Folgenden ist das Beispiel erläutert mit den dazugehörigen Kriterien, welche dadurch erfüllt werden. In Abbildung 4.1 ist die textuelle Beschreibung der Choreographie visuell dargestellt mittels einem erweiterteten BPMN 2.0 Kollaborationsdiagramm, da das Kriterium „(K7) Partnerübergreifender Datenfluss“ in normalen BPMN 2.0 Kollaborationsdiagrammen nicht unterstützt wird.

- Die Choreographie besteht aus drei Teilnehmern: $A$, $B$ und $C$ (K2).
- Teilnehmer $A$ schickt zunächst eine Anfrage $Request\ 1$ an $B$. Im Anschluss daran bekommt er ein Datenobjekt $Data\ Object\ 1$ von $B$ zugesendet (K5), (K7). Anschließend wartet $A$ bis zur Frist $Deadline$, welche hier im abstrakten Beispiel nicht genauer spezifiziert ist, auf eine Antwort von Teilnehmer $B$ namens $Reply\ 1$. Falls diese Nachricht innerhalb der Frist ankommt, führt Teilnehmer $A$ noch die Aktivität $Activity\ A01$ aus, bevor er in den zweiten Endzustand $End\ 2$ gelangt. Falls die Frist überschritten wird, gelangt $A$ direkt in den Endzustand $End\ 1$ (K3).
- Teilnehmer $B$ nimmt zunächst die Anfrage $Request\ 1$ entgegen. Im Anschluss sendet er das Datenobjekt $Data\ Object\ 1$ an $A$. Anschließend sendet er eine Anfrage an Teilnehmer $C$ namens $Request\ 2$ (K1) und wartet anschließend für unbegrenzte Zeit, auf Antwort $Reply\ 2.1$ oder $Reply\ 2.2$ von Teilnehmer $C$. Falls Antwort $Reply\ 2.1$ gesendet wurde, führt $B$ noch die Aktivität $Activity\ B01$ aus, bevor er die Antwort $Reply\ 1$ an Teilnehmer $A$ sendet. Bekommt $B$ die Antwort $Reply\ 2.2$, wird direkt die Antwort $Reply\ 1$ an $A$ geschickt.
- Teilnehmer $C$ nimmt zunächst die Anfrage $Request\ 2$ von $B$ entgegen. Anschließend wird die Aktivität $Activity\ C01$ ausgeführt, welche ein Datenobjekt namens $Data\ Object\ 2$ als Output hat. Danach wird eine Daten-basierte Abfrage gemacht, welche in unserem Beispiel abfragt, ob $X > Y$ (K4). Falls der Return-Wert der Abfrage $True$ ist, wird die Aktivität $Send\ Reply\ 2.2$ ausgeführt, welche als Input das Datenobjekt $Data\ Object\ 2$ hat (K6). Bei dieser Aktivität wird eine Nachricht an $B$ gesendet. Falls der Return-Wert $False$ ist, wird die Nachricht $Reply\ 2.1$ an Teilnehmer $B$ geschickt.

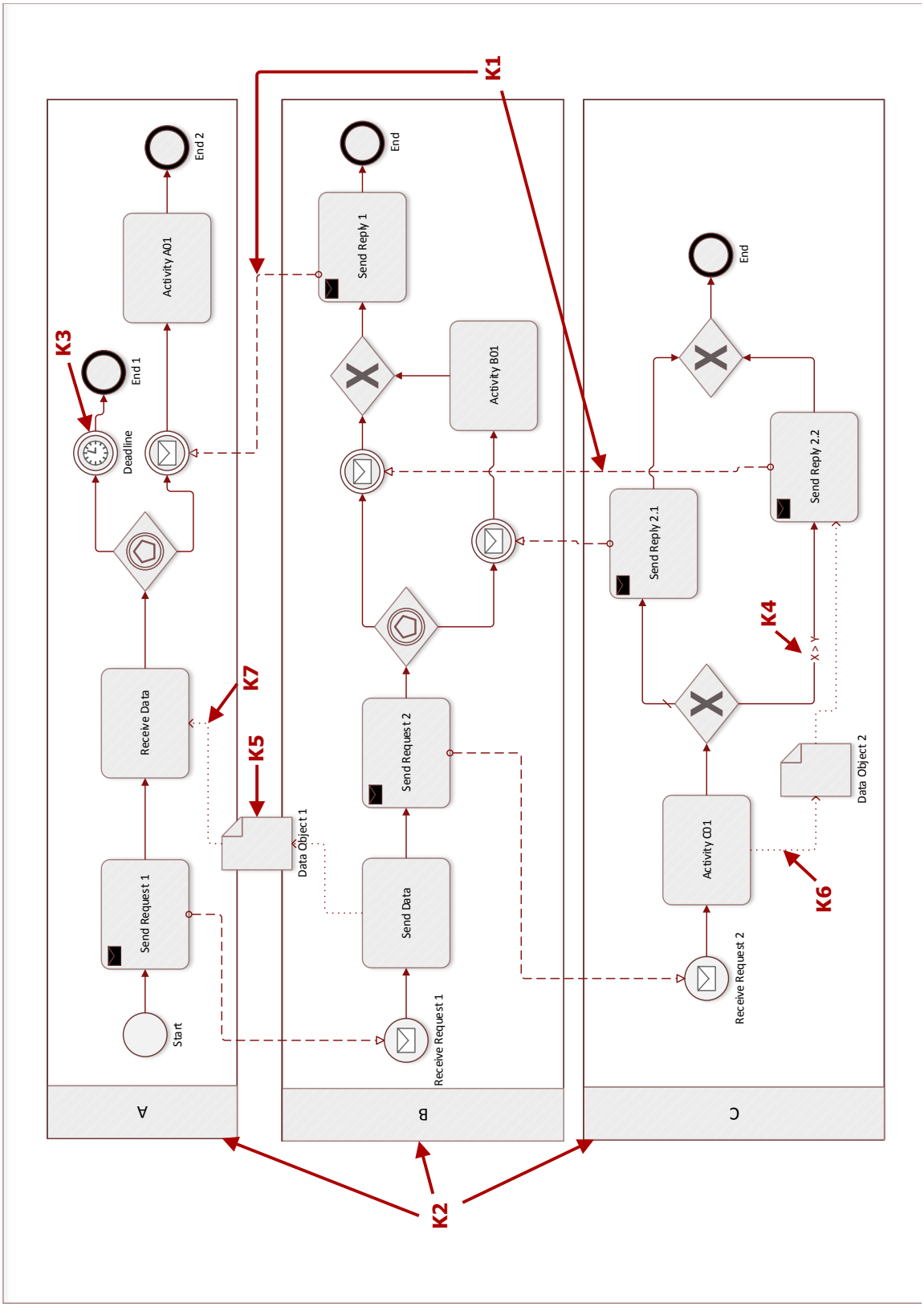



**Abbildung 4.1:** Abstraktes Beispiel einer Choreographie

**Hinweis** Falls kein partnerübergreifender Datenfluss unterstützt wird, aber dennoch Daten ausgetauscht werden können per Nachrichten, so wird es dann dementsprechend auch modelliert. Falls gar keine Daten zwischen Teilnehmern ausgetauscht werden können, so wird hier anstelle dessen ein normaler Nachrichtenaustausch modelliert. Falls Abzweigungen modelliert, aber keine Daten-basierte Abfragen spezifiziert werden können, so werden dennoch die Abzweigungen modelliert, aber es wird dementsprechend erwähnt, dass die Auswahl des Pfades nicht Daten-basiert geschehen kann. Falls in einer Choreographiesprache gar keine Abzweigungen modelliert werden können, so wird automatisch der $True$-Pfad aus unserem Beispiel modelliert. Falls das erste Kriterium zutrifft und es sich nicht schön zeigen lässt durch mehrere Pfeile zu mulitplen Interaktionen, wird dieses Kriterium bei der ersten Interaktion zwischen Teilnehmer $B$ und $C$, $Request2$, markiert.

## 4.3 Analyse der ausgewählten Sprachen

Im Folgenden werden die ausgewählten Sprachen aus Abschnitt 3.4 analysiert und mit Beispielen veranschaulicht.

### 4.3.1 Business Process Model and Notation (BPMN)

Der Standard „Business Process Model and Notation“ (kurz: BPMN) wurde von der „Object Management Group“ (kurz: OMG), welche von mehreren großen und bekannten Firmen (unter anderem HP und IBM) gegründet wurde, im Jahre 2006 entwickelt. Ursprünglich wurde der Standard „Business Process Modeling Notation“ genannt. BPMN dient zur graphischen Modellierung von Geschäftsprozessen. Es existieren bereits mehrere Versionen des BPMN Standards, die auch unterschiedliche Diagrammtypen bereitstellen [Whi04].

BPMN 2.0 ist eine neue Version von BPMN aus dem Jahre 2011, wobei BPMN in der 2. Version umbenannt wurde zu „Business Process Model and Notation“. Ebenfalls wurden hier auch weitere Diagrammtypen hinzugefügt. Die alten Versionen von BPMN werden bei der Analyse weggelassen, da die neuste Version das breiteste Spektrum an Modelliermöglichkeiten bietet. Im Dezember 2013 wurde die neueste Version, BPMN 2.0.2, veröffentlicht, wobei die Änderungen gegenüber Version 2.0 sehr gering sind und diese für unsere Analyse keinen Unterschied machen.

#### BPMN 2.0 Kollaborationsdiagramm

Wie bereits erwähnt, gibt es in BPMN 2.0 mehrere Diagrammtypen. Das Kollaborationsdiagramm ist hierbei eine direkte Erweiterung der in BPMN 1.0 bis 1.2 entwickelten Diagramme [Mod11].

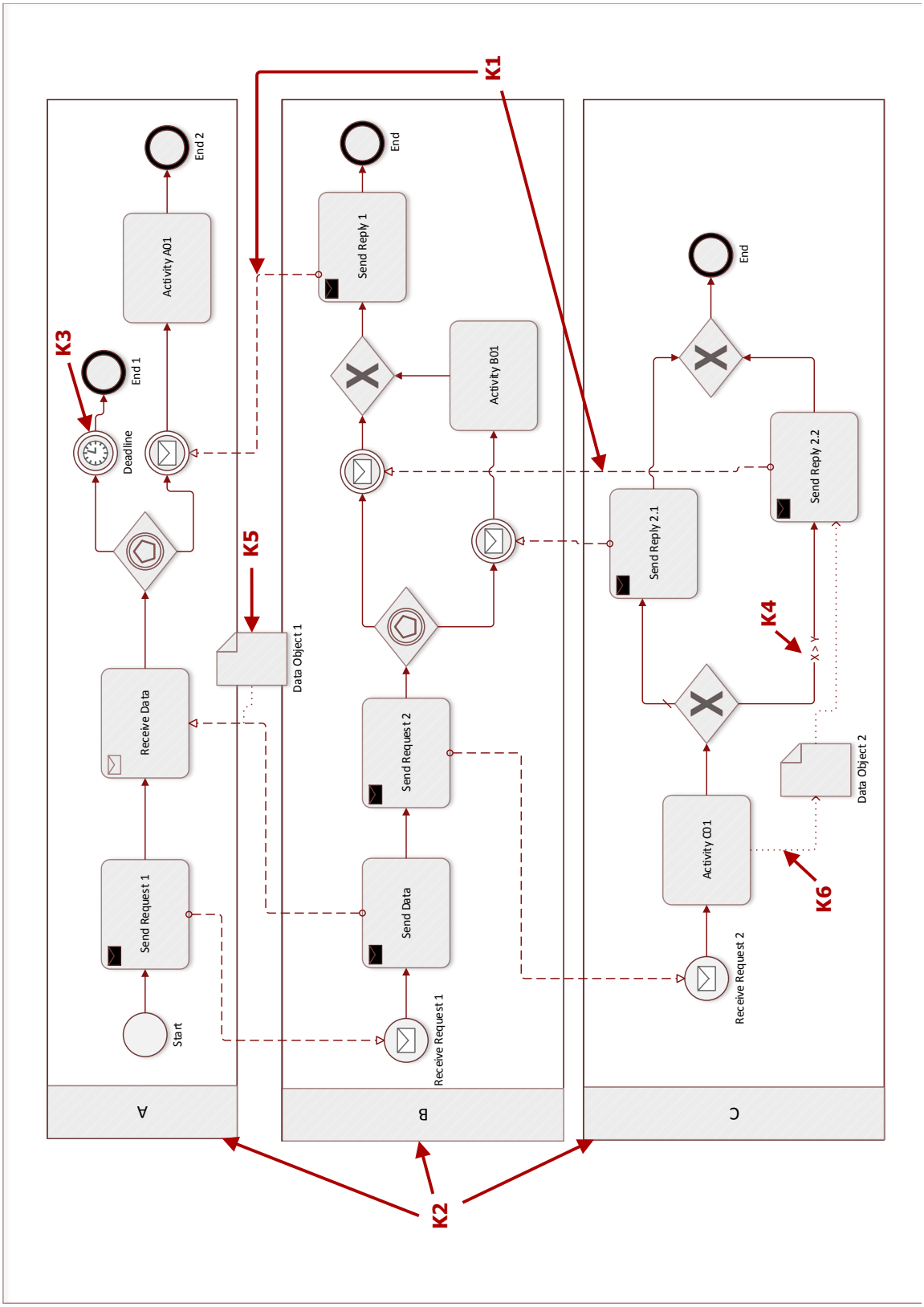



**Abbildung 4.2:** Umsetzung der Choreographie aus Abschnitt 4.2 mit einem BPMN 2.0 Kollaborationsdiagramm

Im Folgenden wird die Eignung von BPMN 2.0 Kollaborationsdiagrammen zur Choreographiemodellierung mittels der in Abschnitt 4.1 vorgestellten Kriterien evaluiert. In Abbildung 4.2 ist die Choreographie aus Abschnitt 4.2 mittels einem BPMN 2.0 Kollaborationsdiagramm umgesetzt.

| K1 | K2 | K3 | K4 | K5 | K6 | K7 |
|---|---|---|---|---|---|---|
| ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | × |

- *(K1) Multilaterale Interaktionen*: Ja, die Teilnehmer werden in BPMN 2.0 Kollaborationsdiagrammen über sogenannte „Pools“ definiert. Die Interaktionen zwischen diesen Pools geschehen durch „Receive“- und „Send-Tasks“. Es gibt hierbei keine Einschränkungen für die Anzahl der Pools und die Kommunikationen zwischen den beliebigen Pools.
- *(K2) Teilnehmertopologie*: Ja, die Teilnehmertopologie wird durch die Pools erreicht. In BPMN 2.0 Kollaborationsdiagrammen ist es auch möglich die minimale und maximale Teilnehmeranzahl eines Typs über die Attribute der Pools anzugeben.
- *(K3) Zeit-Constraints*: Ja, durch „Timer-Events“ können Zeit-Constraints modelliert werden, wobei bei den Timer-Events dann die dazu nötige Information beschrieben ist.
- *(K4) Daten-basierter Kontrollfluss*: Ja, es werden exklusive Gateways angeboten, bei welchem man Daten-basierte Abfragen tätigen kann. Der Pfad welcher die angegebenen Bedingungen erfüllt, wird schließlich ausgewählt.
- *(K5) Modellierung von Datenobjekten*: Ja, es können Datenobjekte explizit modelliert werden.
- *(K6) Expliziter Datenfluss*: Ja, die Datenobjekte können innerhalb eines Pools durch einen expliziten Datenfluss als Input oder Output für einzelne Aktivitäten modelliert werden.
- *(K7) Partnerübergreifender Datenfluss*: Nein, da die Pools untereinander Daten nur via Nachrichten austauschen können, muss man die Datenobjekte an Nachrichten anhängen, um sie an den anderen Pool zu übermitteln. Einen direkten partnerübergreifenden Datenfluss gibt es somit nicht.

### BPMN 2.0 Konversationsdiagramm

Ein BPMN 2.0 Konversationsdiagramm ist eine vereinfachte Form von einem Kollaborationsdiagramm, welches jedoch die gleichen Möglichkeiten zur Modellierung hat [Mod11]. Deshalb sind alle Kriterien die auf Kollaborationsdiagramme zutreffen, ebenfalls zutreffend für Konversationsdiagramme.

| K1 | K2 | K3 | K4 | K5 | K6 | K7 |
|---|---|---|---|---|---|---|
| ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | × |

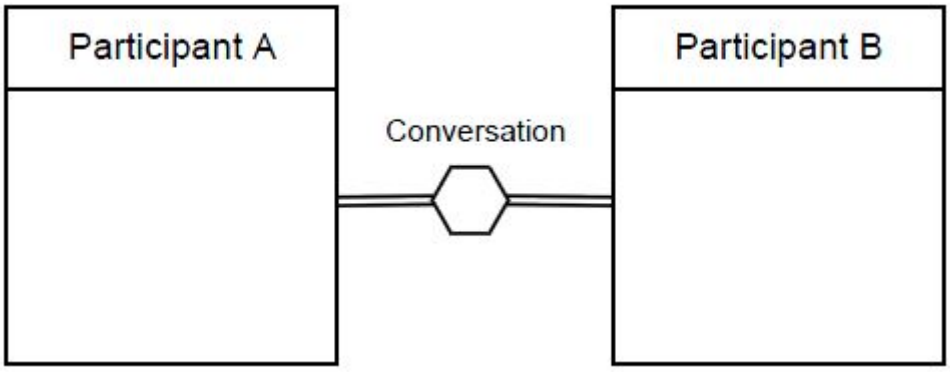



**Abbildung 4.3:** Abstraktes Beispiel von einem BPMN 2.0 Konversationsdiagramm

Konversationsdiagramme haben noch mit verschiedenen „Conversation Nodes“ und „Conversation Links“ zusätzliche Elemente, mithilfe deren man die Konversationen zwischen den Pools vereinfacht und übersichtlicher darstellen kann, da nicht jeder Pool ein Send- bzw. Receive-Task benötigt. Die Prozesse der einzelnen Teilnehmer werden im Normalfall nicht modelliert, was dazu führen würde, dass *(K3) Zeit-Constraints, (K4) Daten-basierter Kontrollfluss, (K5) Modellierung von Datenobjekten und (K6) Expliziter Datenfluss* nicht erfüllt sind. Allerdings dürfen eben auch Prozesse mit Aktivitäten, etc. modelliert werden. Da dies erlaubt ist, sind die gleichen Kriterien erfüllt. In Abbildung 4.3 ist ein kleines abstraktes Beispiel mit den neuen Elementen von Konversationsdiagrammen. Hier gibt es zwei Teilnehmer: $Participant\ A$ und $Participant\ B$, welche eine Konversation namens $Conversation$ miteinander führen.

### BPMN 2.0 Choreographiediagramm

Wie bereits erwähnt wurden in BPMN 2.0 zusätzliche Diagrammtypen vorgestellt. Ein BPMN 2.0 Choreographiediagramm ist ein neuer Diagrammtyp davon, welcher wie der Name schon sagt, extra zur Modellierung von Choreographien entwickelt wurde [Mod11].

Im Folgenden wird die Eignung von BPMN 2.0 Choreographiediagrammen zur Choreographiemodellierung mittels der in Abschnitt 4.1 vorgestellten Kriterien evaluiert. In Abbildung 4.4 ist die Choreographie aus Abschnitt 4.2 mittels einem BPMN 2.0 Choreographiediagramm umgesetzt.

| K1 | K2 | K3 | K4 | K5 | K6 | K7 |
|---|---|---|---|---|---|---|
| ✓ | × | ✓ | ✓ | × | × | × |

- *(K1) Multilaterale Interaktionen*: Ja, in BPMN 2.0 Choreographiediagrammen werden lediglich die Interaktionen zwischen den verschiedenen Teilnehmern mittels „Choreography Tasks“ modelliert. Hierbei werden bei jeder Interaktion die daran teilhabenden

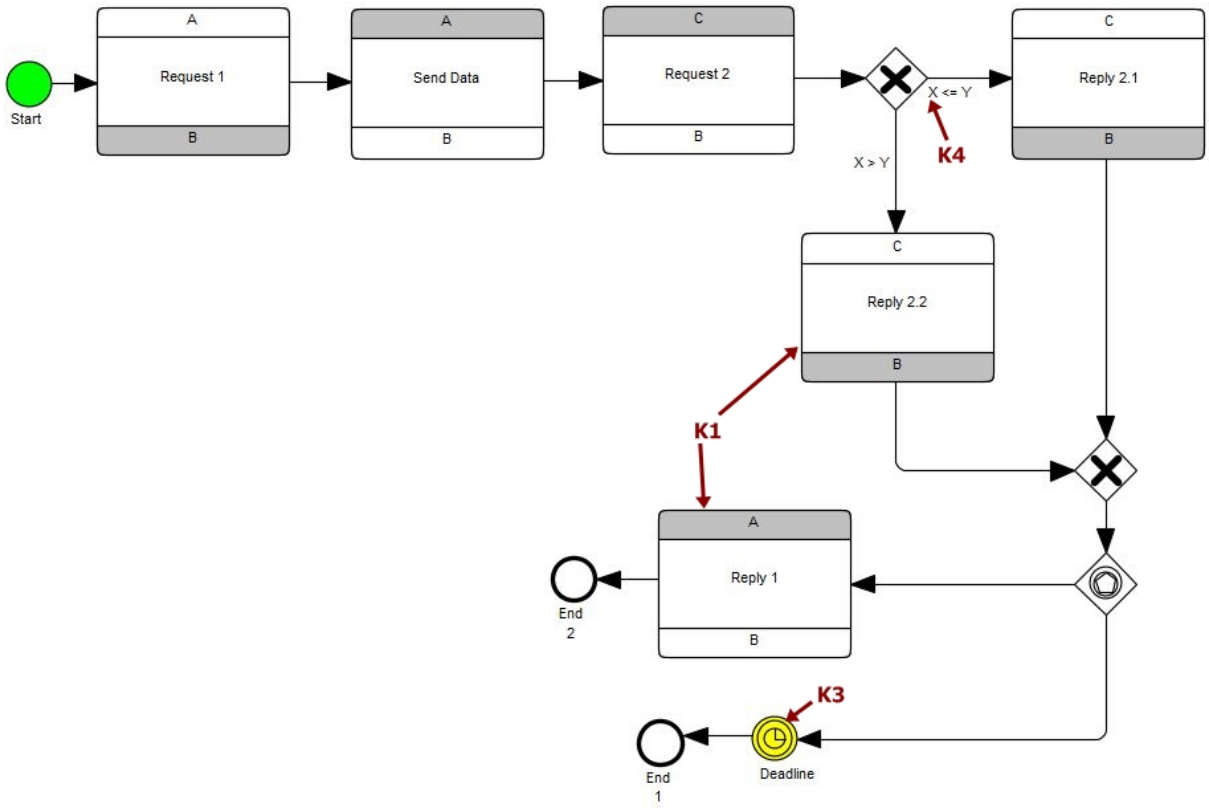



**Abbildung 4.4:** Umsetzung der Choreographie aus Abschnitt 4.2 mit einem BPMN 2.0 Choreographiediagramm

Teilnehmer entweder als Initiator (weiß hinterlegt) oder als Empfänger (grau hinterlegt) angegeben. Es können beliebige Teilnehmer als Initiator und Empfänger angegeben werden. Deshalb wird dieses Kriterium unterstützt.

- *(K2) Teilnehmertopologie*: Nein, die Teilnehmer werden nur implizit bei den einzelnen Choreography-Tasks angegeben. Deshalb gibt es keine Übersicht über alle Teilnehmer in einem Prozess.
- *(K3) Zeit-Constraints*: Ja, BPMN 2.0 Choreographiediagramme erlauben, wie Kollaborationsdiagramme, das Verwenden von „Timer-Events“.
- *(K4) Daten-basierter Kontrollfluss*: Ja, auch in BPMN 2.0 Choreographiediagrammen können exklusive Gateways modelliert werden, bei denen Daten-basierte Abfragen getätigt werden. Allerdings wird in Choreographiediagrammen der Austausch von Daten nicht modelliert, sondern dieser kann nur implizit über die einzelnen Nachrichten geschehen. Eine Bedingung für die Daten-basierten Abfragen ist also, dass die benötigten Daten vor den Gateways per Nachrichten ausgetauscht worden sind, sodass jeder Teilnehmer, der

durch die Entscheidung direkt beeinflusst wird, entweder die benötigten Daten gesendet oder empfangen hat.

- *(K5) Modellierung von Datenobjekten*: Nein, in Choreographiediagrammen wird die Modellierung von Datenobjekten nicht unterstützt.
- *(K6) Expliziter Datenfluss*: Nein, ein expliziter Datenfluss kann in BPMN 2.0 Choreographiediagrammen nicht modelliert werden.
- *(K7) Partnerübergreifender Datenfluss*: Nein, ein partnerübergreifender Datenfluss wird ebenfalls nicht unterstützt.

**Hinweis:** Choreographiediagramme lassen sich in BPMN 2.0 in Kollaborationsdiagramme einbinden.

### 4.3.2 Unified Modeling Language (UML)

Die „Unified Modeling Language“ (kurz: UML) ist eine Modellierungssprache, welche ursprünglich von Booch, Jacobson und Rumbaugh entwickelt wurde [BJR+96] und später dann von der „Object Management Group“ (kurz: OMG) aufgenommen und weiterentwickelt wurde. Es gibt bereits einige Versionen zu UML. Die Entwicklung hatte bereits im Jahre 1994 begonnen, 1997 wurde schließlich die offizielle Version 1.0 veröffentlicht, nachdem sie mithilfe der OMG Gruppe standardisiert worden ist. Nach ein paar kleineren Änderungen und daraus resultierenden Versionen 1.1 bis 1.5, wurde mit UML 2.0 im Jahre 2005 die erste große veränderte Version veröffentlicht [RBJ05]. Im Jahre 2015 wurde die bis heute neueste Version 2.5 publiziert. Eine Übersicht aller UML-Spezifikationen findet sich auf der Webseite der OMG[1]. Wie bereits in BPMN, gibt es auch hier für UML unterschiedliche Diagramme. Im Folgenden wird für die Analyse der verschiedenen Konzepte die neueste UML Version 2.5 verwendet.

#### UML Sequenzdiagramm

Eine Modellierungsart von UML ist das sogenannte Sequenzdiagramm, in welchem die verschiedenen Teilnehmer per Nachrichtenaustausch interagieren [RBJ05].

Im Folgenden wird die Eignung von UML Sequenzdiagrammen zur Choreographiemodellierung mittels der in Abschnitt 4.1 vorgestellten Kriterien evaluiert. In Abbildung 4.5 ist die Choreographie aus Abschnitt 4.2 mittels einem UML Sequenzdiagramm umgesetzt. Die Notation dieses Sequenzdiagrammes ist die Notation aus UML 2.0, welche sich gegenüber 2.5 allerdings nicht inhaltlich unterscheidet. Allerdings wären die Balken bei den einzelnen Lifelines in 2.5 nicht vorhanden.

[1] http://www.omg.org/spec/UML/

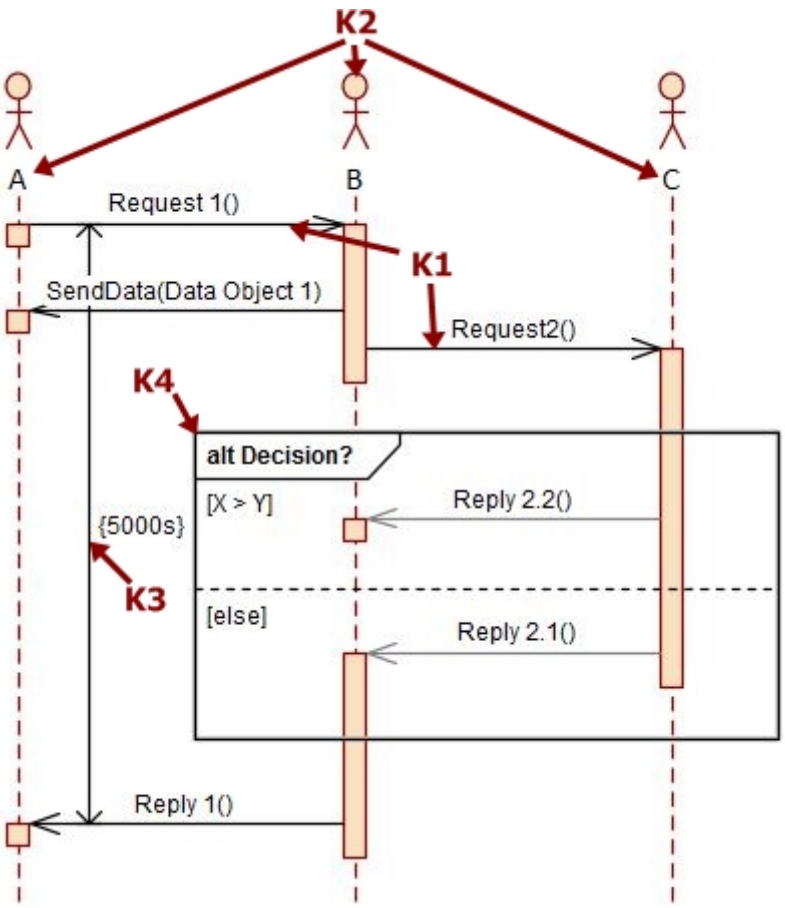



**Abbildung 4.5:** Umsetzung der Choreographie aus Abschnitt 4.2 mit einem UML Sequenzdiagramm

| K1 | K2 | K3 | K4 | K5 | K6 | K7 |
|---|---|---|---|---|---|---|
| ✓ | ✓ | ✓ | ✓ | × | × | × |

- *(K1) Multilaterale Interaktionen*: Ja, es können beliebig viele Teilnehmer modelliert werden, welche untereinander interagieren. Eine Begrenzung gibt es hier nicht. Die verschiedenen Teilnehmer werden in UML Sequenzdiagrammen als sogenannte „Lifelines“ repräsentiert, wobei dort zusätzlich zum Namen der Teilnehmer noch eine vertikale Linie nach unten geht, welche eine Zeitachse darstellen soll, wobei dort dann die Interaktionen zwischen den Teilnehmern modelliert sind. Diese können untereinander durch Nachrichten miteinander interagieren.
- *(K2) Teilnehmertopologie*: Ja, die Teilnehmertopologie wird über die Lifelines definiert.
- *(K3) Zeit-Constraints*: Ja, mithilfe der Konstrukte „DurationConstraint“ und „TimeConstraint“ lassen sich zeitliche Beschränkungen modellieren. Hierbei wird beim Senden einer Nachricht an einen anderen Teilnehmer ein Timer gestartet. Innerhalb der vorgegebenen

Zeit wartet man dann schließlich (wobei man auch noch andere Interaktionen beiläufig durchführen kann) auf eine Antwort des anderen Teilnehmers, da die Teilnehmer eben nur per Nachrichtenaustausch interagieren. In unserem Beispiel haben wir $5000\ s$ als DurationConstraint angegeben.

- *(K4) Daten-basierter Kontrollfluss*: Ja, man kann durch den Konstrukt „CombinedFragment“ alternative Pfade modellieren, basierend auf Daten-abhängigen Abfragen. Man kann auch durch das „Loop CombinedFragment“ z.B. verschiedene Aktionen wiederholt ausführen, bis sich z.B. die abgefragte Variable ändert, wobei daraufhin die Schleife verlassen wird.
- *(K5) Modellierung von Datenobjekten*: Nein, es wird keine Modellierung von Datenobjekten unterstützt.
- *(K6) Expliziter Datenfluss*: Nein, es kann in UML Sequenzdiagrammen kein expliziter Datenfluss modelliert werden.
- *(K7) Partnerübergreifender Datenfluss*: Nein, ein partnerübergreifender Datenfluss kann nicht modelliert werden. Der Datenaustausch geschieht implizit über Nachrichten.

### UML Zeitdiagramm

Ein UML Zeitdiagramm ist eine spezielle Form von UML Sequenzdiagrammen, wobei der zeitliche Aspekt hier eine ganz zentrale Rolle spielt. Während bei Sequenzdiagrammen die Zeitdimension implizit durch die Länge der „Lifelines“ ausgedrückt werden, gibt es hier eine genaue Zeitachse (die $x$-Achse). Deshalb befinden sich die Lifelines hier nicht nebeneinander, mit dem Teilnehmer und einer vertikalen Linie, sondern werden übereinander repräsentiert mit einer horizontalen Linie. Ebenfalls gibt es eine „state lifeline“ oder „condition lifeline“, durch welche z.B. verschiedene Status einer Rolle oder eines Objekts dargestellt werden können [RBJ05].

Im Folgenden wird die Eignung von UML Zeitdiagrammen zur Choreographiemodellierung mittels der in Abschnitt 4.1 vorgestellten Kriterien evaluiert. In Abbildung 4.6 ist ein Zeitdiagramm angegeben, wobei ein Benutzer (*Web User*), eine Website aufrufen möchte.

| K1 | K2 | K3 | K4 | K5 | K6 | K7 |
|---|---|---|---|---|---|---|
| ✓ | ✓ | ✓ | × | × | × | × |

- Die Kriterien *(K1) Multilaterale Interaktionen*, *(K2) Teilnehmertopologie*, *(K3) Zeit-Constraints*, *(K5) Modellierung von Datenobjekten*, *(K6) Expliziter Datenfluss* und *(K7) Partnerübergreifender Datenfluss* unterscheiden sich nicht von den bereits vorgestellten UML Sequenzdiagrammen. Die Umsetzungen dieser Kriterien, sofern eine Umsetzung möglich ist, sind äquivalent.

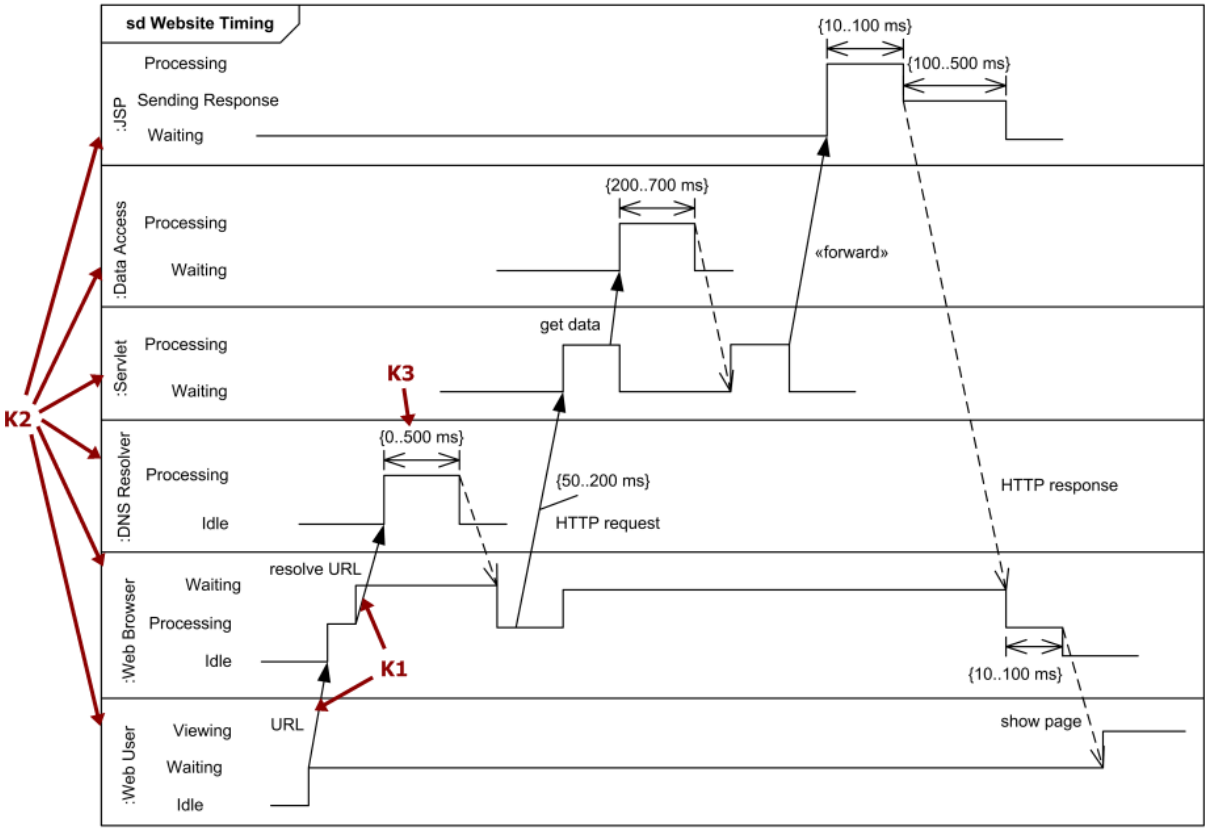



**Abbildung 4.6:** Beispiel einer Choreographie mit einem UML Zeitdiagramm[2]

- *(K4) Daten-basierter Kontrollfluss*: Nein, in UML Zeitdiagrammen wird das Konstrukt „CombinedFragment“ nicht angeboten, sowie auch kein anderes Konstrukt mithilfe dessen man dieses Konstrukt ersetzen könnte. Somit können keine Daten-basierten Abfragen getätigt werden, welche den Kontrollfluss beeinflussen.

Wie man hier sieht ist, dass UML Zeitdiagramme basierend auf unseren Kriterien schlechter abschneiden als UML Sequenzdiagramme. Dies liegt daran, dass ein Daten-basierter Kontrollfluss hier nicht möglich ist. Obwohl UML Sequenzdiagramme den zeitlichen Aspekt nicht im Vordergrund haben, können dort dennoch Zeit-Constraints modelliert werden.

[2]http://www.uml-diagrams.org/website-latency-uml-timing-diagram-example.html

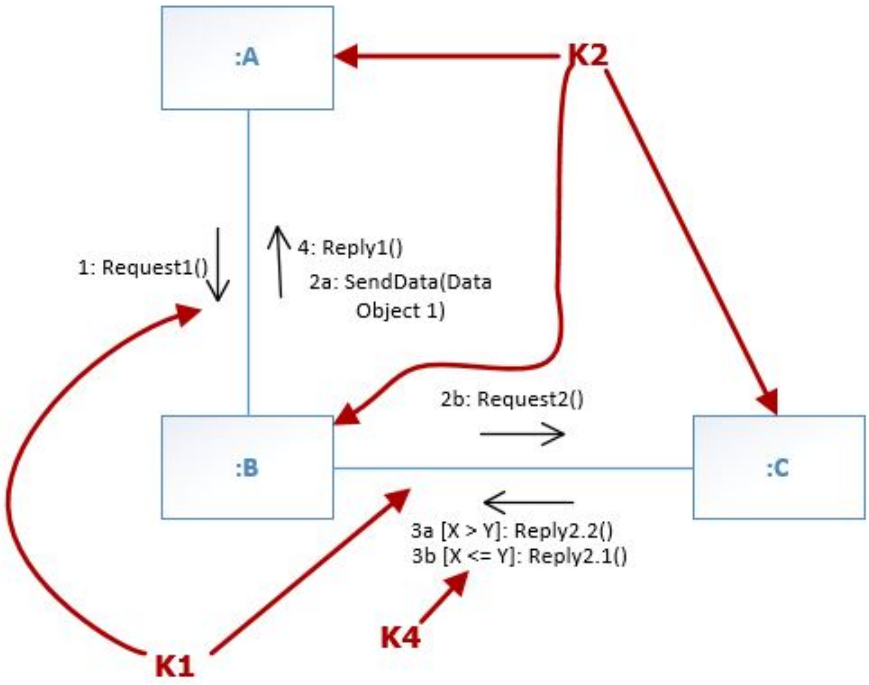



**Abbildung 4.7:** Umsetzung der Choreographie aus Abschnitt 4.2 mit einem UML Kommunikationsdiagramm

### UML Kommunikationsdiagramm

UML Kommunikationsdiagramme sind eine weitere Form von Interaktionsdiagrammen von UML. Kommunikationsdiagramme konzentrieren sich hier ausschließlich auf den Nachrichtenaustausch der einzelnen Teilnehmer. Sie werden als vereinfachte Sequenzdiagramme angesehen [RBJ05].

Im Folgenden wird die Eignung von UML Kommunikationsdiagrammen zur Choreographiemodellierung mittels der in Abschnitt Abschnitt 4.1 vorgestellten Kriterien evaluiert. In Abbildung 4.7 ist die Choreographie aus Abschnitt 4.2 mittels einem UML Kommunikationsdiagramm umgesetzt.

| K1 | K2 | K3 | K4 | K5 | K6 | K7 |
|---|---|---|---|---|---|---|
| ✓ | ✓ | × | ✓ | × | × | × |

- *(K1) Multilaterale Interaktionen*: Ja, multilaterale Interaktionen werden in UML Kommunikationsdiagrammen ebenfalls unterstützt. Auch hier werden die einzelnen Teilnehmer als

„Lifelines“ repräsentiert, welche untereinander beliebig viele Nachrichten austauschen können. Die graphische Modellierung unterscheidet sich jedoch von den Repräsentationen in Sequenzdiagrammen. Hier erfolgt die Modellierung ähnlich wie bei einem Graphen. Die zeitliche Abfolge der Interaktionen ist hier deshalb nicht auf dem ersten Blick erkennbar. Diese erfolgt durch die Kantenbeschriftungen, in welchen durchnummeriert ist wann eine Nachricht gesendet wird. Nachrichten können auch parallel gesendet werden. Um diese einzelnen Nachrichten zu unterscheiden, wird hinter der Zahl dann noch ein zusätzlicher eigener Buchstabe angegeben.

- *(K2) Teilnehmertopologie*: Ja, die Teilnehmertopologie wird auch hier durch die Lifelines spezifiziert.
- *(K3) Zeit-Constraints*: Nein, Konstrukte zur Modellierung von Zeit-Constraints werden hier nicht unterstützt.
- *(K4) Daten-basierter Kontrollfluss*: Ja, auch in UML Kommunikationsdiagrammen lassen sich Daten-basierte Konditionen modellieren, welche erfüllt sein müssen um den entsprechenden Pfad zu gehen. Diese werden direkt vor der Nachricht als „Guard“ angegeben.
- *(K5) Modellierung von Datenobjekten*: Nein, wie bereits in den anderen Interaktionsdiagrammen, welche UML anbietet, können auch hier Datenobjekte nicht explizit modelliert werden.
- *(K6) Expliziter Datenfluss*: Nein, es lässt sich kein Datenfluss in UML Kommunikationsdiagrammen modellieren.
- *(K7) Partnerübergreifender Datenfluss*: Nein, die Modellierung eines partnerübergreifenden Datenflusses wird nicht unterstützt.

### UML Aktivitätsdiagramm

UML Aktivitätsdiagramme gehören im Gegensatz zu den anderen hier vorgestellten Modellierungsarten aus UML nicht zu den Interaktionsdiagrammen, sondern zu den Verhaltensdiagrammen [RBJ05]. Dennoch wurden sie in verschiedenen Arbeiten [BDO06; KP06; MH08] als geeignete Sprache zur graphischen Modellierung von Choreographien verwendet.

Im Folgenden wird die Eignung von UML Aktivitätsdiagrammen zur Choreographiemodellierung mittels der in Abschnitt 4.1 vorgestellten Kriterien evaluiert. In Abbildung 4.8 ist die Choreographie aus Abschnitt 4.2 mittels einem UML Aktivitätsdiagramm umgesetzt.

| K1 | K2 | K3 | K4 | K5 | K6 | K7 |
|---|---|---|---|---|---|---|
| ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

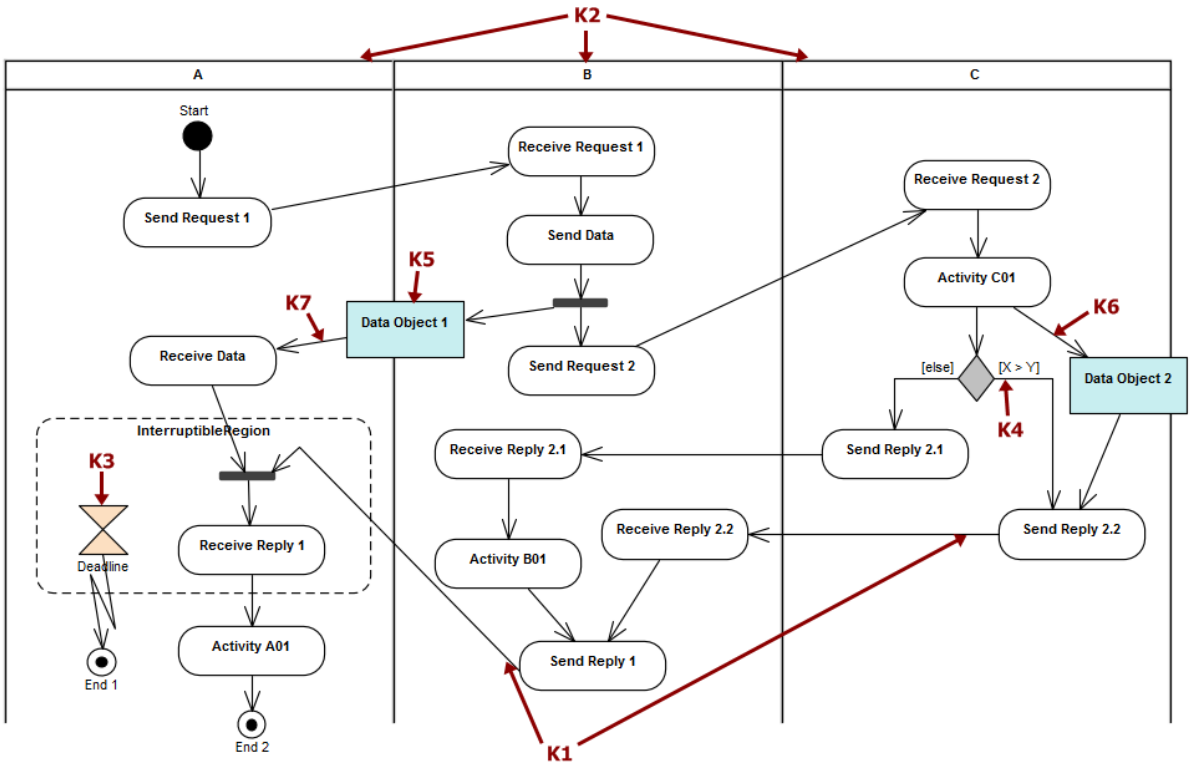



**Abbildung 4.8:** Umsetzung der Choreographie aus Abschnitt 4.2 mit einem UML Aktivitätsdiagramm

- *(K1) Multilaterale Interaktionen*: Ja, UML Aktivitätsdiagramme erlauben die Modellierung von verschiedenen Teilnehmern via „Partitionen“. In UML Aktivitätsdiagrammen werden außerdem die einzelnen Aktivitäten der Partitionen modelliert. Ein Kontrollfluss sorgt dabei für die Reihenfolge der ausgeführten Aktivitäten. Der Start einer Choreographie wird durch ein sogenanntes „Initial Node“ Element gekennzeichnet und das Ende einer Choreographie wird durch ein „Activity Final Node“ Element repräsentiert. Es können mehrere Enden modelliert werden, die Choreographie wird dabei durch das erste ausgelöste End-Element beendet. Es lassen sich außerdem mehrere Start-Elemente modellieren, was dazu führt, dass verschiedene Kontrollflüsse nebenläufig gestartet werden. Durch die Datenflüsse werden Interaktionen direkt unterstützt.
- *(K2) Teilnehmertopologie*: Ja, durch die Modellierung von Partitionen wird eine Teilnehmertopologie gewährleistet.
- *(K3) Zeit-Constraints*: Ja, man kann zeitliche Bedingungen durch sogenannte „Accept Time Event Actions“ modellieren. Wenn Timeouts innerhalb der „Interruptible Regions“ ausgelöst werden, wird dann der entsprechende Timeout-Pfad gelaufen.

- *(K4) Daten-basierter Kontrollfluss*: Ja, man kann durch „Decision Nodes“ den nachfolgenden Kontrollfluss durch bestimmte Abfragen beeinflussen. Damit kann man Abfragen machen, ob bisher alle wichtigen Daten vorhanden sind z.B., falls nicht wird ein alternativer Pfad abgelaufen.
- *(K5) Modellierung von Datenobjekten*: Ja, die Datenobjekte werden als „Object Nodes“ modelliert und agieren hierbei als Input und/oder Output für einzelne Aktivitäten.
- *(K6) Expliziter Datenfluss*: Ja, es gibt einen expliziten Datenfluss, welcher von der Repräsentation her dem Kontrollfluss gleicht. Diese Datenobjekte dienen als Input und/oder Output von einzelnen Aktivitäten.
- *(K7) Partnerübergreifender Datenfluss*: Ja, in UML Aktivitätsdiagrammen macht es keinen Unterschied, ob man Daten innerhalb einer Partition oder auch an eine Aktivität einer anderen Partition verschickt. Deshalb ist dieses Kriterium erfüllt.

### UML Interaktionsübersichtsdiagramm

UML Interaktionsübersichtsdiagramme zählen zu den Interaktionsdiagrammen. Hier werden Interaktionen durch eine Variante von Aktivitätsdiagrammen definiert, um eine Übersicht über den Kontrollfluss zu erhalten [RBJ05].

Im Folgenden wird die Eignung von UML Interaktionsübersichtsdiagrammen zur Choreographiemodellierung mittels der in Abschnitt 4.1 vorgestellten Kriterien evaluiert. In Abbildung 4.9 ist die Choreographie aus Abschnitt 4.2 mittels einem UML Interaktionsübersichtsdiagramm umgesetzt. Zur graphischen Unterstützung der Evaluierung der einzelnen Kritieren, ist in Abbildung 4.9 die Choreographie aus Abschnitt 4.2 mittels einem UML Interaktionsübersichtsdiagramm umgesetzt/modelliert.

| K1 | K2 | K3 | K4 | K5 | K6 | K7 |
|---|---|---|---|---|---|---|
| ✓ | × | ✓ | ✓ | × | × | × |

- *(K1) Multilaterale Interaktionen*: Ja, multilaterale Interaktionen werden unterstützt. Die Modellierung ist wie eine Mischung aus Aktivitäts- und Sequenzdiagrammen. Wie schon Aktivitätsdiagramme gibt es hier den „Initial Node“, welcher den Anfang signalisiert und einen (oder mehrere) „Final Node(s)“. Der Kontrollfluss gleicht ebenfalls dem von Aktivitätsdiagrammen. Ein Unterschied gegenüber Aktivitätsdiagrammen ist hierbei, dass es hier keine Partitionen gibt. Die Teilnehmer werden implizit in den einzelnen „Interactions“ modelliert. Hier wird auch die Interaktion zwischen den Teilnehmern in Form von Sequenzdiagrammen genau modelliert. Eine Begrenzung für die Anzahl der Teilnehmer oder Interaktionen gibt es nicht.
- *(K2) Teilnehmertopologie*: Nein, eine Teilnehmertopologie wird nicht unterstützt, da die Teilnehmer nur implizit bei den einzelnen Interaktionen angegeben werden.

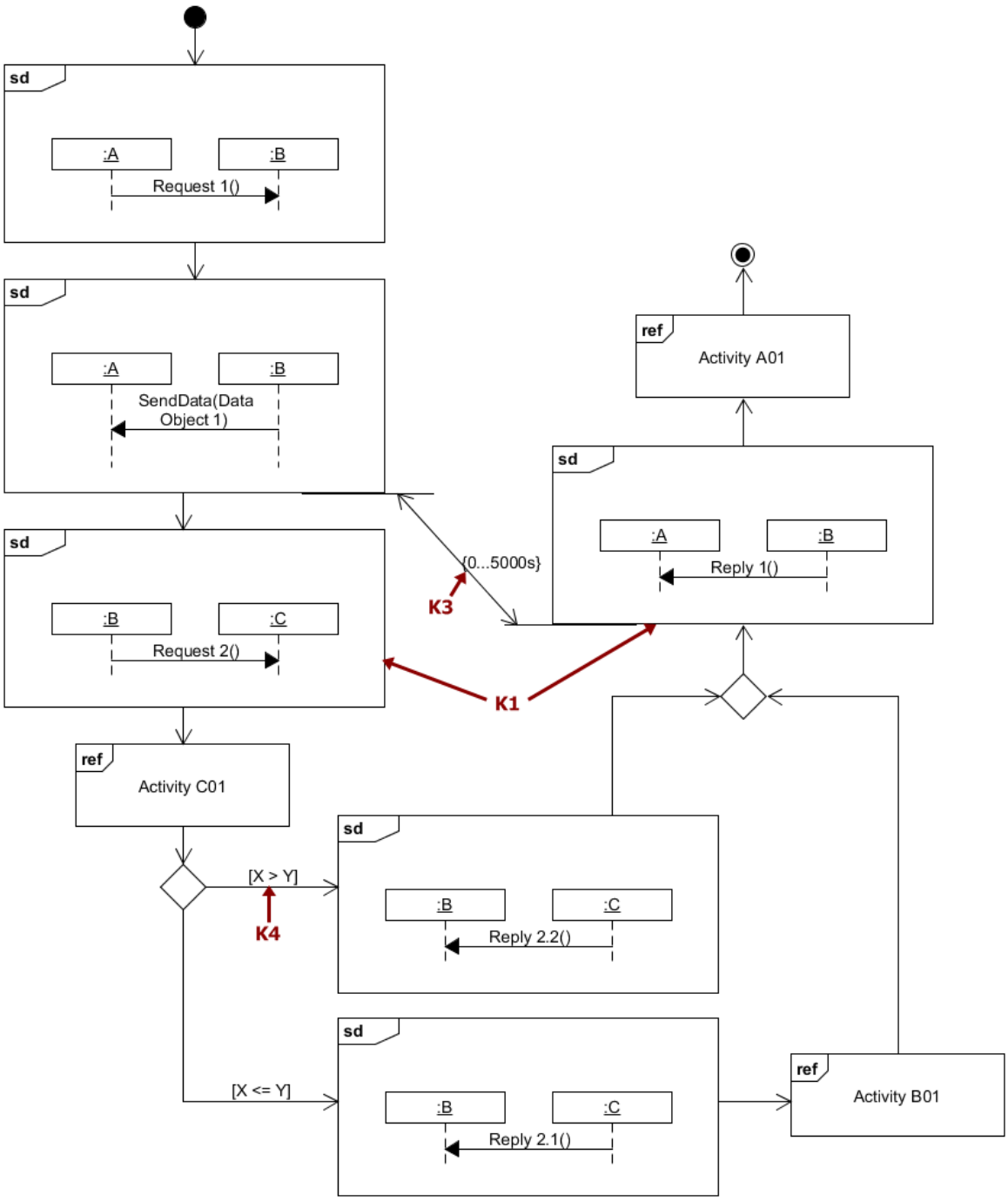



**Abbildung 4.9:** Umsetzung der Choreographie aus Abschnitt 4.2 mit einem UML Interaktionsübersichtsdiagramm

- *(K3) Zeit-Constraints*: Ja, die zeitliche Beschränkungen erfolgen durch sogenannte „DurationConstraints“ oder „TimeConstraints“, welche modelliert werden wie in Sequenzdiagrammen. In unserem Beispiel wählen wir die Zeitbegrenzung so, dass nach Abschluss der $Send\ Data$ Interaktion innerhalb von $5000s$ die Interaktion $Reply1$ durchgeführt werden muss.
- *(K4) Daten-basierter Kontrollfluss*: Ja, der Daten-basierte Kontrollfluss erfolgt, wie schon in Aktivitätsdiagrammen mittels „Decision Nodes“.
- *(K5) Modellierung von Datenobjekten*: Nein, im Gegensatz zu UML Aktivitätsdiagrammen lassen sich in Interaktionsübersichtsdiagrammen keine Datenobjekte modellieren.
- *(K6) Expliziter Datenfluss*: Nein, ein expliziter Datenfluss existiert in UML Interaktionsübersichtsdiagrammen nicht.
- *(K7) Partnerübergreifender Datenfluss*: Nein, ein partnerübergreifender Datenfluss existiert in UML Interaktionsübersichtsdiagrammen nicht. Die Daten müssen impliziten über Nachrichten ausgetauscht werden.

### 4.3.3 Web Service Flow Language (WSFL)

Die „Web Service Flow Language“ wurde von Leymann [Ley+01] im Jahre 2001 entworfen und gilt somit als eine der älteren Sprachen, welche zur Choreographiemodellierung geeignet sind. WSFL ist eine XML-basierte Sprache zur Beschreibung von Web Service Strukturen. WSFL ist außerdem ein Vorgänger der „Web Services Business Process Execution Language“ (kurz: BPEL; [ACD+03]), ein Vorreiter zur Modellierung von Orchestrierungen.

Im Folgenden wird die Eignung von WSFL zur Choreographiemodellierung mittels der in Abschnitt 4.1 vorgestellten Kriterien evaluiert. In Listing 4.1 ist die Choreographie aus Abschnitt 4.2 mittels WSFL umgesetzt. Aus Gründen der Übersichtlichkeit wurden die Definition der `Messages`, sowie der `<portType>`- und `<serviceProviderType>`-Konstrukte weggelassen.

| K1 | K2 | K3 | K4 | K5 | K6 | K7 |
|---|---|---|---|---|---|---|
| ✓ | ✓ | × | ✓ | ✓ | ✓ | × |

- *(K1) Multilaterale Interaktionen*: Ja, WSFL bietet zwei Modelle: „Flow Model“ und „Global Model“. In einem Flow-Modell wird der Kontrollfluss eines Prozesses definiert. Im globalen Modell wird beschrieben, wie die verschiedenen Web Services untereinander interagieren. Die einzelnen Aktivitäten werden im Flow-Modell definiert. Außerdem werden auch hier einzelne Interaktionen modelliert. Im globalen Modell hat man mithilfe der `<plugLink>`-Konstrukte eine globale Sicht auf die einzelnen Interaktionen. Eine Beschränkung für die Anzahl der Teilnehmer und Interaktionen gibt es hierbei nicht.
- *(K2) Teilnehmertopologie*: Ja, die Teilnehmertopologie wird im globalen Model definiert.

- *(K3) Zeit-Constraints*: Nein, WSFL bietet keine Konstrukte zur Modellierung von Zeit-Constraints an.
- *(K4) Daten-basierter Kontrollfluss*: Ja, in WSFL gibt es sogenannte „Transition Conditions“, welche den Kontrollfluss durch Daten-basierte Abfragen leiten. Dies wird im Flow-Modell modelliert.
- *(K5) Modellierung von Datenobjekten*: Ja, in WSFL werden Daten als Nachrichten definiert.
- *(K6) Expliziter Datenfluss*: Ja, der Datenaustausch geschieht über `<dataLink>`-Konstrukte. Bei diesen Datalinks werden die Quell- und Ziel-Aktivitäten mit angegeben.
- *(K7) Partnerübergreifender Datenfluss*: Nein, die `<dataLink>`-Konstrukte können nur innerhalb eines Flow-Modells angegeben werden. Somit kann kein partnerübergreifender Datenfluss garantiert werden.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <definitions targetNamespace="http://www.example.org/example-choreography/"
4               xmlns:tio="http://www.example.org/example-choreography/"
5               xmlns="http://schemas.xmlsoap.org/wsfl/"
6               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
7               xsi:schemaLocation="http://schemas.xmlsoap.org/wsfl/ wsfl-schema.xsd">
8
9      <flowModel name="AFlow"
10                serviceProviderType="tio:TypeA">
11
12         <serviceProvider name="B" type="tio:TypeB"/>
13
14
15         <activity name="issueRequest1">
16             <performedBy serviceProvider="B"/>
17             <implement>
18                 <export>
19                     <target portType="tio:ARequester"
20                             operation="SendRequest1"/>
21                 </export>
22             </implement>
23         </activity>
24
25         <activity name="getData">
26             <performedBy serviceProvider="B"/>
27             <implement>
28                 <export>
29                     <target portType="tio:ARequester"
30                             operation="ReceiveData"/>
31                 </export>
32             </implement>
```

```
33          </ activity >
34
35          < activity  name="getReply1">
36              <performedBy serviceProvider="B"/>
37              <implement>
38                  <export>
39                      < target  portType="tio:ARequester"
40                                operation="ReceiveReply1"/>
41                  </export>
42              </implement>
43          </ activity >
44
45          < activity  name="ActivityA01">
46              <performedBy serviceProvider="A"/>
47              <implement>
48                  < internal >
49                      ...
50                  </ internal >
51              </implement>
52          </ activity >
53
54          <controlLink  source="issueRequest1"
55                        target ="getData"/>
56          <controlLink  source="getData"
57                        target ="getReply1"/>
58          <controlLink  source="getReply1"
59                        target ="ActivityA01"/>
60      </flowModel>
61
62      <flowModel name="BFlow"
63                 serviceProviderType ="tio:TypeB">
64
65
66          < serviceProvider  name="A" type="tio:TypeA"/>
67          < serviceProvider  name="C" type="tio:TypeC"/>
68
69          <export  lifecycleAction ="spawn">
70              < target  portType="tio:BReplier"
71                        operation="ReceiveRequest1"/>
72          </export>
73
74          < activity  name="issueData">
75              <performedBy serviceProvider="A"/>
76              <implement>
77                  <export>
78                      < target  portType="tio:BReplier"
79                                operation="SendData"/>
80                  </export>
81              </implement>
82          </ activity >
83
```

```
84          < activity  name="issueRequest2">
85              <performedBy serviceProvider="C"/>
86              <implement>
87                  <export>
88                      < target  portType="tio:BRequester"
89                                operation="SendRequest2"/>
90                  </export>
91              </implement>
92          </ activity >
93
94          < activity  name="getReply21">
95              <performedBy serviceProvider="C"/>
96              <implement>
97                  <export>
98                      < target  portType="tio:BRequester"
99                                operation="ReceiveReply2.1"/>
100                 </export>
101             </implement>
102         </ activity >
103
104         < activity  name="getReply22">
105             <performedBy serviceProvider="C"/>
106             <implement>
107                 <export>
108                     < target  portType="tio:BRequester"
109                               operation="ReceiveReply2.2"/>
110                 </export>
111             </implement>
112         </ activity >
113
114         < activity  name="ActivityB01">
115             <performedBy serviceProvider="B"/>
116             <implement>
117                 < internal >
118                     ...
119                 </ internal >
120             </implement>
121         </ activity >
122
123         < activity  name="issueReply1">
124             <performedBy serviceProvider="A"/>
125             <implement>
126                 <export>
127                     < target  portType="tio:BReplier"
128                               operation="SendReply1"/>
129                 </export>
130             </implement>
131         </ activity >
132
133         <controlLink  source="issueData"
134                       target="issueRequest2"/>
```

```
135         <controlLink source="issueRequest2"
136                      target="getReply21"/>
137         <controlLink source="issueRequest2"
138                      target="getReply22"/>
139         <controlLink source="getReply21"
140                      target="ActivityB01"/>
141         <controlLink source="ActivityB01"
142                      target="issueReply1"/>
143         <controlLink source="getReply22"
144                      target="issueReply1"/>
145     </flowModel>
146
147     <flowModel name="CFlow"
148                serviceProviderType="tio:TypeC">
149
150         <serviceProvider name="B" type="tio:TypeB"/>
151
152         <export lifecycleAction="spawn">
153             <target portType="tio:CReplier"
154                     operation="ReceiveRequest2"/>
155
156         </export>
157
158         <activity name="ActivityC01">
159             <performedBy serviceProvider="C"/>
160             <implement>
161                 <internal>
162                     ...
163                 </internal>
164
165             </implement>
166         </activity>
167
168         <activity name="issueReply21">
169             <performedBy serviceProvider="C"/>
170             <implement>
171                 <export>
172                     <target portType="tio:CReplier"
173                             operation="SendReply2.1"/>
174                 </export>
175             </implement>
176         </activity>
177
178         <activity name="issueReply22">
179             <performedBy serviceProvider="C"/>
180             <implement>
181                 <export>
182                     <target portType="tio:CReplier"
183                             operation="SendReply2.2"/>
184                 </export>
185             </implement>
```

```
186         </ activity >
187
188         <controlLink  source="ActivityC01"
189                       target ="issueReply21"
190                       transitionCondition ="X<=Y"/>
191         <controlLink  source="ActivityC01"
192                       target ="issueReply22"
193                       transitionCondition ="X>Y"/>
194         <dataLink name="DataObject2Transfer"
195                    source="ActivityC01"
196                    target ="issueReply22"/>
197     </flowModel>
198
199     <globalModel name="Example-ChoreographyGlobal"
200                  serviceProviderType ="exampleChoreography">
201
202         <serviceProvider  name="A" type="tio:TypeA"/>
203         <serviceProvider  name="B" type="tio:TypeB"/>
204         <serviceProvider  name="C" type="tio:TypeC"/>
205
206         <plugLink>
207             <source  serviceProvider ="A"
208                      portType="tio:ARequester"
209                      operation="SendRequest1"/>
210             <target  serviceProvider ="B"
211                      portType="tio:BReplier"
212                      operation="ReceiveRequest1"/>
213         </plugLink>
214
215         <plugLink>
216             <source  serviceProvider ="B"
217                      portType="tio:BReplier"
218                      operation="SendData"/>
219             <target  serviceProvider ="A"
220                      portType="tio:ARequester"
221                      operation="ReceiveData"/>
222         </plugLink>
223
224         <plugLink>
225             <source  serviceProvider ="B"
226                      portType="tio:BRequester"
227                      operation="SendRequest2"/>
228             <target  serviceProvider ="C"
229                      portType="tio:CReplier"
230                      operation="ReceiveRequest2"/>
231         </plugLink>
232
233         <plugLink>
234             <source  serviceProvider ="C"
235                      portType="tio:CReplier"
236                      operation="SendReply2.1"/>
```

```
237             <target  serviceProvider ="B"
238                      portType="tio:BRequester"
239                      operation="ReceiveReply2.1"/>
240         </plugLink>
241
242         <plugLink>
243             <source  serviceProvider ="C"
244                      portType="tio:CReplier"
245                      operation="SendReply2.2"/>
246             <target  serviceProvider ="B"
247                      portType="tio:BRequester"
248                      operation="ReceiveReply2.2"/>
249         </plugLink>
250
251         <plugLink>
252             <source  serviceProvider ="B"
253                      portType="tio:BReplier"
254                      operation="SendReply1"/>
255             <target  serviceProvider ="A"
256                      portType="tio:ARequester"
257                      operation="ReceiveReply1"/>
258         </plugLink>
259     </globalModel>
260
261 </ definitions >
```

**Listing 4.1:** Umsetzung der Choreographie aus Abschnitt 4.2 in WSFL

### 4.3.4 Event-driven Process Chain (EPC)

„Event-driven Process Chain“ (kurz: EPC) wurden ursprünglich von Keller, Scheer und Nüttgens [KSN92] im Jahres 1992 unter dem deutschen Namen „Ereignisgesteuerte Prozessketten“ (kurz: EPK) entworfen. Über die Jahre wurden diese EPCs erweitert. Eine neuere Version, bei welchem unter anderem A.W. Scheer [STA05], einer der Entwickler der ersten Version, beteiligt war wurde im Jahr 2005 präsentiert. Die folgende Analyse bezieht sich deshalb auf die neuere Version.

Im Folgenden wird die Eignung von EPCs zur Choreographiemodellierung mittels der in Abschnitt 4.1 vorgestellten Kriterien evaluiert. In Abbildung 4.10 ist die Choreographie aus Abschnitt 4.2 mittels EPC umgesetzt.

| K1 | K2 | K3 | K4 | K5 | K6 | K7 |
|---|---|---|---|---|---|---|
| ✓ | × | × | ✓ | ✓ | ✓ | ✓ |

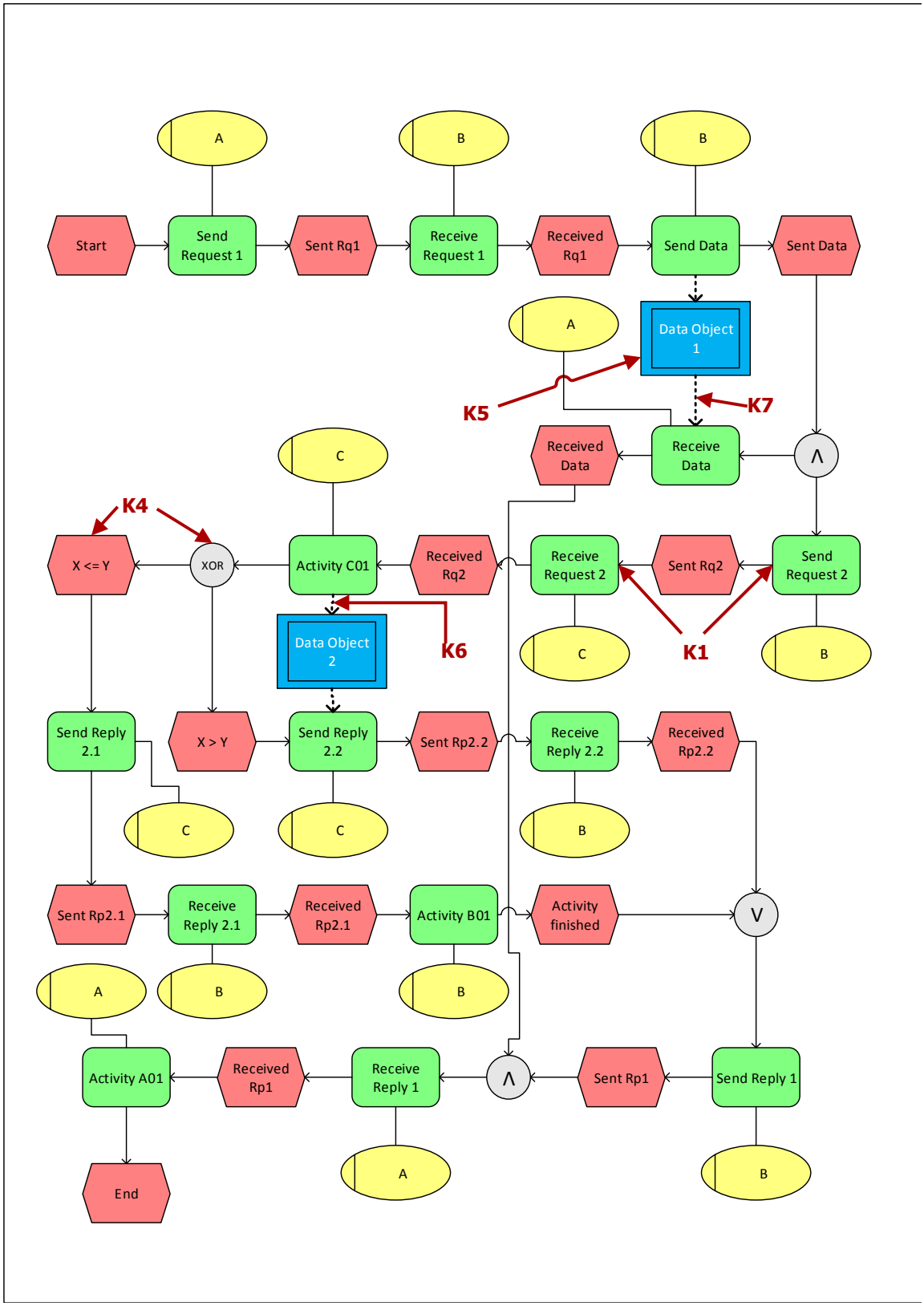



**Abbildung 4.10:** Umsetzung der Choreographie aus Abschnitt 4.2 mit EPC

- *(K1) Multilaterale Interaktionen*: Ja, in einem EPC werden die einzelnen Aktivitäten, sowie Events in einem Workflow modelliert. Jeder Aktivität kann man die zuständige Organisation bzw. Organisationen zuordnen, wodurch die Choreographie entsteht. Diese Organisationen sind die einzelnen Teilnehmer. Durch diesen Kontrollfluss und den Datenfluss, sowie den Angaben der Organisationen bei jeder Aktivität, lassen sich die Interaktionen zwischen den einzelnen Organisationen modellieren. Aus Gründen der Übersichtlichkeit, ist dieses Kriterium im Beispiel in Abbildung 4.10 nicht gekennzeichnet. Es ist aber ersichtlich, dass multilaterale Interaktionen unterstützt und hier im Modell auch wiedergegeben werden.
- *(K2) Teilnehmertopologie*: Nein, die Organisationen werden implizit bei den Aktivitäten angegeben. Somit wird die Teilnehmertopologie in EPC nicht unterstützt.
- *(K3) Zeit-Constraints*: Nein, es werden keine Konstrukte zur Modellierung von Zeit-Constraints angeboten.
- *(K4) Daten-basierter Kontrollfluss*: Ja, durch den XOR-Connector in Verbindung mit Konditionen als Event-Angabe lässt sich ein Daten-basierter Kontrollfluss modellieren.
- *(K5) Modellierung von Datenobjekten*: Ja, es lassen sich Daten in EPC modellieren.
- *(K6) Expliziter Datenfluss*: Ja, die modellierten Daten werden als Input und/oder Output für die einzelnen Aktivitäten angegeben.
- *(K7) Partnerübergreifender Datenfluss*: Ja, in EPC macht es keinen Unterschied, welcher Teilnehmer welche Aktivitäten ausführt. So kann ein partnerübergreifender Datenfluss modelliert werden, indem der Output einer Aktivität eines Teilnehmers gleichzeitig der Input einer anderen Aktivität eines anderen Teilnehmers ist.

### 4.3.5 Web Services Choreography Description Language (WS-CDL)

Die „Web Sevices Choreography Description Language“ (kurz: WS-CDL) ist eine von Barros, Dumas und Oaks [BDO05] definierte XML-basierte Choreographiesprache, welche sich aus den Vorgängern „Web Services Choreography Interface“ (kurz: WSCI; [AAF+02]) und „Web Service Conversation Language“ (kurz: WSCL; [BBB+02]) entwickelt hat und diese um zusätzliche Elemente erweitert.

Im Folgenden wird die Eignung von WS-CDL zur Choreographiemodellierung mittels der in Abschnitt 4.1 vorgestellten Kriterien evaluiert. In Listing 4.2 ist die Choreographie aus Abschnitt 4.2 mittels WS-CDL umgesetzt. Dieses Beispiel ist nicht vollständig spezifiziert, da die `channelVariable`-Elemente in den `<interaction>`-Blöcken auf Variablen zugreifen, welche hier aus Gründen der Übersichtlichkeit nicht definiert werden. Diese Elemente sind allerdings irrelevant für die Analyse der Kriterien.

| K1 | K2 | K3 | K4 | K5 | K6 | K7 |
|---|---|---|---|---|---|---|
| ✓ | ✓ | ✓ | ✓ | ✓ | × | × |

- *(K1) Multilaterale Interaktionen*: Ja, WS-CDL erlaubt die Modellierung von mehreren Rollentypen mittels `<roleType>`-Konstrukten, welche die Teilnehmer repräsentieren. Die Interaktionen werden mittels `<interaction>`-Konstrukten spezifiziert. Eine Einschränkung für die Anzahl der Rollen oder Interaktionen untereinander gibt es in WS-CDL nicht.
- *(K2) Teilnehmertopologie*: Ja, in WS-CDL werden alle vorkommenden Rollen durch das Konstrukt `<roleType>` aufgezählt, womit sich eine Topologie ergibt.
- *(K3) Zeit-Constraints*: Ja, man kann mithilfe von `<timeout>`-Konstrukten eine Zeit für die Interaktionen festlegen, welche eingehalten werden muss.
- *(K4) Daten-basierter Kontrollfluss*: Ja, WS-CDL bietet Konstrukte, bei welchen man bestimmte Daten abfragen kann und darauf basierend eine Aktion ausgeführt oder nicht ausgeführt wird. Dies geschieht innerhalb von `<workunit>`-Konstrukten, bei welchen man eine `guard`-Kondition angeben kann, welche erfüllt sein muss, sodass der entsprechende Block ausgeführt wird.
- *(K5) Modellierung von Datenobjekten*: Ja, es können in WS-CDL Variablen definiert werden, in welchen Daten gespeichert werden können. Dies geschieht über `variable`-Definitionen innerhalb des `<variableDefinitions>`-Konstruktes.
- *(K6) Expliziter Datenfluss*: Nein, es kann kein expliziter Datenfluss in WS-CDL modelliert werden.
- *(K7) Partnerübergreifender Datenfluss*: Nein, der Austausch von Daten muss über Nachrichten geschehen.

### Beispiel

```
1  <?xml version ="1.0"  encoding="UTF-8"?>
2  <schema xmlns="http://www.w3.org/2001/XMLSchema"
3          xmlns:cdl="http :// www.w3.org/2005/10/cdl"
4          targetNamespace="http :// www.w3.org/2005/10/cdl"
5          elementFormDefault="qualified ">
6
7    <package name="Example"
8             author="Nico_L&amp;#228;ssig"
9             targetNamespace="http :// www.example.org/example-choreography/"
10            xmlns="http :// www.w3.org/2005/10/cdl"
11            xmlns:tns="http :// www.example.org/example-choreography/">
12
13      <roleType name="A">  (K2)
14          <behavior name="Behavior1"/>
15      </roleType>
```

```
16      <roleType name="B">
17          <behavior name="Behavior2"/>
18      </roleType>
19      <roleType name="C">
20          <behavior name="Behavior3"/>
21      </roleType>
22      <relationshipType name="A2B">
23          <roleType typeRef="tns:A"/>
24          <roleType typeRef="tns:B"/>
25      </relationshipType>
26      <relationshipType name="B2C">
27          <roleType typeRef="tns:B"/>
28          <roleType typeRef="tns:C"/>
29      </relationshipType>
30
31      <choreography name="ExampleChoreography">
32
33          <relationship type="tns:A2B"/>
34          <relationship type="tns:B2C"/>
35
36          <variableDefinitions>    (K5)
37              <variable name="DataObject1" roleTypes="tns:A tns:B"/>
38              <variable name="DataObject2" roleTypes="tns:C"/>
39          </variableDefinitions>
40
41          <sequence>
42              <interaction name="A2BInteraction1"
43                           channelVariable="tns:A2BCV"
44                           operation="Req1RepDat">
45                  <participate relationshipType="tns:A2B"
46                               fromRoleTypeRef="tns:A" toRoleTypeRef="tns:B"/>
47                  <exchange name="Request1"
48                           action="request">
49                      <send> ... </send>
50                      <receive> ... </receive>
51                  </exchange>
52                  <exchange name="DataTransfer"
53                           action="respond">
54                      <send    variable="cdl:getVariable('DataObject1','','')"/>
55                      <receive variable="cdl:getVariable('DataObject1','','')"/>
56                  </exchange>
57              </interaction>
58              <interaction name="B2CInteraction1"    (K1)
59                           channelVariable="tns:B2CCV"
60                           operation="Req2">
61                  <participate relationshipType="tns:B2C"
62                               fromRoleTypeRef="tns:B" toRoleTypeRef="tns:C"/>
63                  <exchange name="Request2"
64                           action="request">
65                      <send> ... </send>
66                      <receive> ... </receive>
```

```
67                  </exchange>
68              </ interaction >
69              <workunit name="Workunit1"
70                        guard="cdl:getVariable('X','','',  'tns:C') > cdl:getVariable('Y','','',
    'tns:C')">  (K4)
71                  <interaction name="B2CInteraction2"
72                               channelVariable="tns:B2CCV"
73                               operation="Rep21">
74                      <participate relationshipType="tns:B2C"
75                                   fromRoleTypeRef="tns:B" toRoleTypeRef="tns:C"/>
76                      <exchange name="Reply2.1"
77                                action="respond">
78                          <send> ... </send>
79                          <receive> ... </receive>
80                      </exchange>
81                  </ interaction >
82              </workunit>
83              <workunit name="Workunit2"
84                        guard="cdl:getVariable('X','','',  'tns:C') <= cdl:getVariable('Y','','',
    'tns:C')">
85                  <interaction name="B2CInteraction3"
86                               channelVariable="tns:B2CCV"
87                               operation="Rep22">
88                      <participate relationshipType="tns:B2C"
89                                   fromRoleTypeRef="tns:B" toRoleTypeRef="tns:C"/>
90                      <exchange name="Reply2.2"
91                                action="respond">
92                          <send> ... </send>
93                          <receive> ... </receive>
94                      </exchange>
95                  </ interaction >
96              </workunit>
97              <interaction name="A2BInteraction2"
98                           channelVariable="tns:A2BCV"
99                           operation="Rep1">
100                 <participate relationshipType="tns:A2B"
101                              fromRoleTypeRef="tns:A" toRoleTypeRef="tns:B"/>
102                 <exchange name="Reply1"
103                           action="respond">
104                     <send> ... </send>
105                     <receive> ... </receive>
106                 </exchange>
107                 <timeout time-to-complete="2018-09-24Z"/> (K3)
108             </ interaction >
109         </sequence>
110     </choreography>
111 </package>
112 </schema>
```

**Listing 4.2:** Umsetzung der Choreographie aus Abschnitt 4.2 in WS-CDL

### 4.3.6 Declarative Service Flow Language (DecSerFlow)

„DecSerFlow“ ist eine von van der Aalst und Pesic [AP06a] definierte graphische Repräsentation zur deklarativen Modellierung von Choreographien. Die Grundlage von DecSerFlow ist hierbei die textuelle Logik-Sprache „Linear Temporal Logic“ (kurz: LTL). Die Hauptidee ist, dass zunächst einmal in der Design-Phase ein graphisches Modell mittels der Notation von DecSerFlow generiert wird. In der anschließenden Mapping-Phase wird das Modell in LTL Formeln transformiert. Diese lassen sich dann in der abschließenden Phase, zur Laufzeit mithilfe von entsprechender Software ausführen, welche den Arbeitsfluss kontrollieren. Die Tools sind dafür geeignet, um Verstöße oder Probleme des Modells zu detektieren und analysieren [AP06a; AP06b]. In [AP06b] findet sich neben der Modellierung eines einzelnen Services auch noch die Modellierung von mehreren Services und die Kommunikation zwischen diesen.

Im Folgenden wird die Eignung von DecSerFlow zur Choreographiemodellierung mittels der in Abschnitt 4.1 vorgestellten Kriterien evaluiert. In Abbildung 4.11 ist eine typische Buchhandel-Choreographie umgesetzt, in welcher ein Kunde ein Buch ausleihen oder kaufen will. Die genaue Spezifikation dieses Beispiels findet sich in [AP06b].

| K1 | K2 | K3 | K4 | K5 | K6 | K7 |
|---|---|---|---|---|---|---|
| ✓ | ✓ | ✓ | ✓ | × | × | × |

- *(K1) Multilaterale Interaktionen*: Ja, multilaterale Interaktionen werden unterstützt. Die einzelnen Teilnehmer, hier Services genannt, werden in dem Beispiel aus Abbildung 4.11 in Form von Boxen dargestellt. Eine spezielle Bezeichnung hierfür wird nicht definiert. Innerhalb der Box werden dann die einzelnen Aktivitäten modelliert. Die Kommunikation zwischen den verschiedenen Teilnehmern werden via Pfeile von einer Aktivität eines Services zu einer Aktivität eines anderen Services symbolisiert. In unserem Beispiel nehmen vier Teilnehmer an einer Choreographie teil: Der Kunde ($customer$), die Buchhandlung ($bookstore$), der Verleger ($publisher$) und der Absender ($shipper$). Die Bedeutung der verschiedenen Pfeil-Typen sind in [AP06a], sowie [AP06b] definiert. In diesem Beispiel wird eine Choreographie aus Sicht des Kunden repräsentiert. Deshalb sind hier der Zusammenhang der einzelnen Aktivitäten nur aus Sicht des Kunden genau modelliert, sowie die Interaktionen zu den anderen Teilnehmern vom Kunden aus und umgekehrt. Wie die anderen Teilnehmer untereinander kommunizieren oder wie deren interner Ablauf aussieht, wird hier deshalb nicht gezeigt.
- *(K2) Teilnehmertopologie*: Ja, die Teilnehmertopologie wird in DecSerFlow durch die einzelnen „Boxen“ für jeden Service bewerkstelligt.
- *(K3) Zeit-Constraints*: Ja, es werden zwar nicht explizit Konstrukte zur Modellierung von Zeit-Constraints angegeben, jedoch wird erwähnt, dass ein Zustand nicht nur eine Aktivität sein muss, sondern auch Informationen, wie Zeit oder Daten enthalten kann. Somit können Informationen bezüglich der Zeit modelliert werden und darauf basierend auch Constraints. Dieses Kriterium wird im Beispiel nicht wiedergegeben.

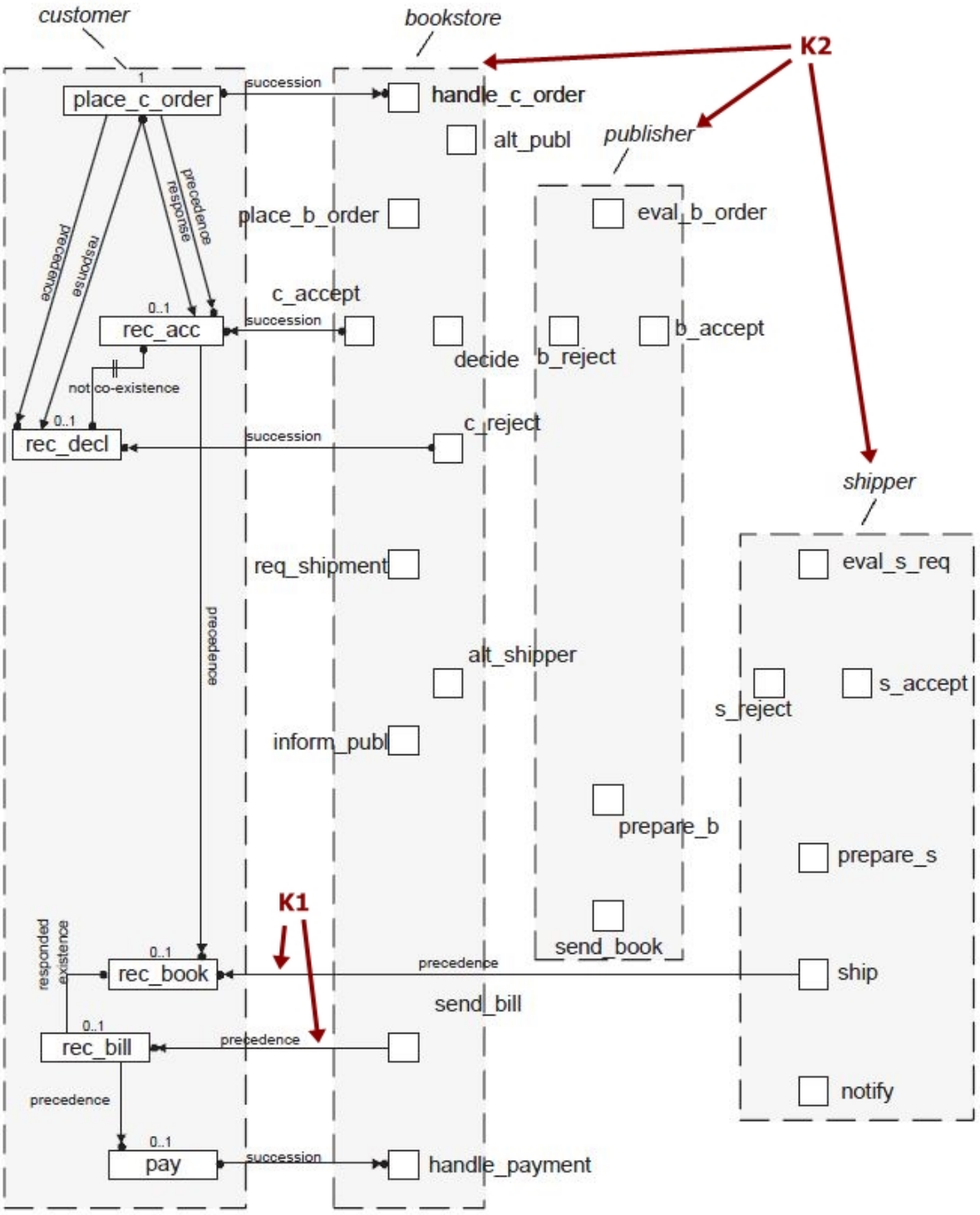



**Abbildung 4.11:** Beispiel einer Choreographie in DecSerFlow aus [AP06b]

- *(K4) Daten-basierter Kontrollfluss*: Ja, Daten-abhängige Abfragen können getätigt werden, woraufhin dann der jeweilige Pfad gewählt wird. Es wird erwähnt, dass die einzelnen Zustände Informationen von Daten enthalten können. Dieses Kriterium wird in diesem Beispiel ebenfalls nicht wiedergegeben.
- *(K5) Modellierung von Datenobjekten*: Nein, Daten können lediglich implizit bei den einzelnen Aktivitäten als zusätzliche Informationen definiert werden. Eine explizite Modellierung von Datenobjekten gibt es nicht.
- *(K6) Expliziter Datenfluss*: Nein, im graphischen Modell von DecSerFlow wird ein Datenfluss nicht spezifiziert.
- *(K7) Partnerübergreifender Datenfluss*: Nein, ein partnerübergreifender Datenfluss kann in DecSerFlow ebenfalls nicht modelliert werden.

### 4.3.7 Global Calculus

Der „Global Calculus“ ist ein formaler Calculus, welcher seine Ursprünge in WS-CDL hat. Der Global Calculus wurde ursprünglich von Carbone et al. [CHY07] im Jahre 2006 veröffentlicht [CHY+06]. Ein Jahr später wurde schließlich ein Paper zu diesem Calculus im Springer-Verlag veröffentlicht.

Im Folgenden wird die Eignung von Global Calculus zur Choreographiemodellierung mittels der in Abschnitt 4.1 vorgestellten Kriterien evaluiert. In Listing 4.3 ist die Choreographie aus Abschnitt 4.2 mittels Global Calculus umgesetzt.

| K1 | K2 | K3 | K4 | K5 | K6 | K7 |
|---|---|---|---|---|---|---|
| ✓ | × | (✓) | ✓ | × | × | × |

- *(K1) Multilaterale Interaktionen*: Ja, multilaterale Interaktionen werden im Global Calculus unterstützt. Es können beliebige Interaktionen zwischen Teilnehmern via sogenannter „Channels“ modelliert werden. Die Teilnehmer werden dabei implizit bei der Interaktion mit angegeben. Aus unserem Beispiel: $A \to B : Ch1(\boldsymbol{\nu}\, s)$ bezeichnet hierbei, dass $A$ ein Service Channel namens $Ch1$ aufgerufen und eine neue Session namens $s$ initiiert hat. $A \to B : s\langle Request1\rangle$ bezeichnet hierbei eine Interaktion zwischen $A$ und $B$ mittels Session $s$, namens $Request1$. Bei einem sequentiellen Ablauf wird ein Punkt zwischen den einzelnen Interaktionen gemacht. **0** steht für eine Inaktion.
- *(K2) Teilnehmertopologie*: Nein, eine Übersicht über alle Teilnehmer kann im Global Calculus nicht angegeben werden. Die Teilnehmer werden bei den einzelnen Interaktionen implizit mit angegeben.

- *(K3) Zeit-Constraints*: Teilweise, es werden keine Konstrukte zur Modellierung von Zeit-Constraints in der Definition von Global Calculus angegeben. In [CHY+06] wird allerdings erwähnt, dass die Modellierung von Timeouts (und diversen anderen Elementen) zwar nicht im Kern von Global Calculus enthalten sind, dies jedoch als Erweiterung möglich ist. Deshalb ist dieses Kriterium teilweise erfüllt.
- *(K4) Daten-basierter Kontrollfluss*: Ja, es können „if-else“ Abfragen getätigt werden, wodurch basierend auf den Daten unterschiedliche Pfade weitergelaufen werden. Somit ist das Kriterium erfüllt. In unserem Beispiel bezeichnet $X > Y@C$ die Daten-basierte Abfrage, wobei die Kondition $X > Y$ ist, welche bei Teilnehmer $C$ liegt.
- *(K5) Modellierung von Datenobjekten*: Nein, Datenobjekte können nicht explizit modelliert werden. Es können zwar sogenannte „Expressions“ per Nachrichten ausgetauscht werden, allerdings werden diese implizit bei den Nachrichten angegeben. In unserem Beispiel wird bei $A \to B : s\langle Request1, 1000, y\rangle$ die Expression $1000$ der Variable $y$, welche in $B$ sein muss, zugewiesen.
- *(K6) Expliziter Datenfluss*: Nein, ein expliziter Datenfluss lässt sich mit dem Global Calculus nicht definieren.
- *(K7) Partnerübergreifender Datenfluss*: Nein, ein partnerübergreifender Datenfluss gibt es ebenfalls nicht im Global Calculus. Es können nur per Nachrichtenaustausch Expressions weitergegeben werden, welche einer bestimmten Variable des Empfängers zugewiesen werden.

```
1  A → B: Ch1(ν s).
2  A → B: s⟨Request1⟩.
3  B → A: s⟨DataTransfer, 1000, y⟩.
4  B → C: Ch2(ν t).                    (K1)
5  B → C: t⟨Request2⟩.
6  if X > Y@C then                     (K4)
7     {C → B: t⟨Reply2.1⟩.
8      B → A: s⟨Reply1⟩. 0}
9  else
10    {C → B: t⟨Reply2.2⟩.
11     B → A: s⟨Reply1⟩. 0}
```

**Listing 4.3:** Umsetzung der Choreographie aus Abschnitt 4.2 mit dem Global Calculus

### 4.3.8 BPEL4Chor

BPEL4Chor ist eine von Decker et al. [DKLW07] im Jahre 2007 vorgestellte Erweiterung der zweiten Version des Standards für Orchestrierungsprachen „Business Process Execution Language“ (kurz: BPEL; [JEA+07]). Diese Erweiterungen wurden entwickelt, um die Modellierung von Choreographien auf Basis von BPEL zu ermöglichen. Für die Erweiterungen wurden drei verschiedene Aspekte betrachtet: (1) Verhaltensbeschreibung der Teilnehmer, (2) Modellierung einer Teilnehmertopologie und (3) Teilnehmer Grounding.

Im Folgenden wird die Eignung von BPEL4Chor zur Choreographiemodellierung mittels der in Abschnitt 4.1 vorgestellten Kriterien evaluiert. In Listing 4.4 ist die Choreographie aus Abschnitt 4.2 mittels BPEL4Chor umgesetzt. Dieses Beispiel ist nicht ganz vollständig, da der `messageType` in den `<messageLink>`-, sowie `<variable>`-Konstrukten auf eine mittels WSDL definierte Nachricht verweist. Diese sind hier nicht genauer definiert, da sie für die Analyse der Kriterien nicht relevant sind.

| K1 | K2 | K3 | K4 | K5 | K6 | K7 |
|---|---|---|---|---|---|---|
| ✓ | ✓ | ✓ | ✓ | ✓ | × | × |

- *(K1) Multilaterale Interaktionen*: Ja, BPEL4Chor unterstützt die Modellierung von mehreren Teilnehmern welche untereinander beliebig kommunizieren können. Bereits in Standard BPEL ist es möglich, jedoch unterscheidet sich die Modellierung hierfür etwas in BPEL4Chor. In BPEL4Chor wird die Kommunikation zwischen Teilnehmern explizit durch ein neu definiertes `<messageLink>`-Konstrukt modelliert, welches eine Nachrichtenkante spezifiziert.
- *(K2) Teilnehmertopologie*: Ja, die Modellierung einer Teilnehmertopologie ist einer der Aspekte, die durch BPEL4Chor ermöglicht werden. Durch das gleichnamige Konstrukt `<topology>` kann eine Teilnehmertopologie modelliert werden. Hier können ebenfalls neben der Modellierung des Teilnehmertyps auch die Anzahl der Teilnehmer eines Teilnehmertyps angegeben werden. Neben den Teilnehmern enthält die Topologie auch die Nachrichtenkanten (messageLinks) zwischen Teilnehmern.
- *(K3) Zeit-Constraints*: Ja, dies ist bereits im Standard BPEL möglich. Es lassen sich zeitliche Ausdrücke in Form einer Dauer oder eine genaue Zeitangabe in Form eines Datums oder Uhrzeit angeben. Mithilfe des `<onAlarm>` Konstrukts lässt sich dann modellieren, was im Falle einer Zeitüberschreitung bzw. Nichteinhaltung der Deadline passiert. In unserem Beispiel haben wir den Termin $2018-12-24T18:00+01:00$ für die Deadline gewählt.
- *(K4) Daten-basierter Kontrollfluss*: Ja, bereits in BPEL ist es möglich durch Daten-basierte Abfragen, den Kontrollfluss zu leiten. Hierfür können `<if>`-Konstrukte, aber auch Konstrukte wie `<while>` oder `<repeatUntil>` verwendet werden. Die letzteren zwei genannten Konstrukte werden verwendet um Aktivitäten mehrfach auszuführen bis eine

entsprechende Bedingung erfüllt ist oder eine gewisse Anzahl an Schleifendurchläufen (Iterationen) ausgeführt wurden.

- *(K5) Modellierung von Datenobjekten*: Ja, Daten können in BPEL und somit auch BPEL4Chor durch das Konstrukt `<variable>` modelliert werden.
- *(K6) Expliziter Datenfluss*: Nein, in BPEL4Chor kann kein expliziter Datenfluss modelliert werden.
- *(K7) Partnerübergreifender Datenfluss*: Nein, Daten können nur per Nachrichten ausgetauscht werden.

```
1  <process name="BehaviorA"
2           xmlns="http :// docs. oasis -open.org/wsbpel/2.0/process/ abstract "
3           xmlns:b4c="http :// www.bpel4chor.org/pbd/20140331"
4           targetNamespace="http :// www.example.org/example-choreography/type-a/"
5           abstractProcessProfile ="http :// www.bpel4chor.org/profile/20140331">
6    < variables >    (K5)
7      < variable name="DataObject1" messageType="msg:DataTransfer"/>
8    </ variables >
9    <sequence>
10     <invoke name="SendRequest1"/>
11     <receive name="ReceiveData" outputVariable="DataObject1"/>
12     <pick>
13       <onMessage name="ReceiveReply1">
14         <opaqueActivity="ActivityA01"/>
15       </onMessage>
16       <onAlarm> (K3)
17         < until >'2018-12-24T18:00+01:00'</until>
18       </onAlarm>
19     </pick>
20   </sequence>
21 </process>
22
23 <process name="BehaviorB"
24          xmlns="http :// docs. oasis -open.org/wsbpel/2.0/process/ abstract "
25          xmlns:b4c="http :// www.bpel4chor.org/pbd/20140331"
26          targetNamespace="http :// www.example.org/example-choreography/type-b/"
27           abstractProcessProfile ="http :// www.bpel4chor.org/profile/20140331">
28   < variables >
29     < variable name="DataObject1" messageType="msg:DataTransfer"/>
30   </ variables >
31   <sequence>
32     <receive name="ReceiveRequest1" createInstance ="yes"/>
33     <invoke name="SendData" inputVariable="DataObject1"/>
34     <invoke name="SendRequest2"/>
35     <pick>
36       <onMessage name="ReceiveReply2.2"/>
37       <onMessage name="ReceiveReply2.1">
```

```
38          <opaqueActivity="ActivityB01"/>
39        </onMessage>
40      </pick>
41      <invoke name="SendReply1"/>
42    </sequence>
43  </process>
44
45  <process name="BehaviorC"
46             xmlns="http:// docs. oasis-open.org/wsbpel/2.0/process/ abstract "
47             xmlns:b4c="http :// www.bpel4chor.org/pbd/20140331"
48             targetNamespace="http:// www.example.org/example-choreography/type-c/"
49             xmlns:var="http :// www.example.org/example-choreography/type-c/"
50              abstractProcessProfile ="http :// www.bpel4chor.org/profile/20140331">
51    <sequence>
52      <receive name="ReceiveRequest2" createInstance="yes"/>
53      <opaqueActivity name="ActivityC01"/>
54      <if>    (K4)
55        <condition>
56          X > Y
57        </condition>
58        <sequence>
59          <invoke name="SendReply2.2"/>
60        </sequence>
61        <else>
62          <sequence>
63            <invoke name="SendReply2.1"/>
64          </sequence>
65        </else>
66      </if>
67    </sequence>
68  </process>
69
70  <topology name="Choreography"  (K2)
71              targetNamespace="http:// www.example.org/buchung"
72              xmlns:a="http :// www.example.org/example-choreography/type-a/"
73              xmlns:b="http :// www.example.org/example-choreography/type-b/"
74              xmlns:c="http :// www.example.org/example-choreography/type-c/">
75    <import
76      namespace="http://www.example.org/messages/"
77      importType="http:// schemas.xmlsoap.org/wsdl/"/>
78    <participantTypes>
79      <participantType name="A"
80        participantBehaviorDescription ="a:BehaviorA"/>
81      <participantType name="B"
82        participantBehaviorDescription ="b:BehaviorB"/>
83      <participantType name="C"
84        participantBehaviorDescription ="c:BehaviorC"/>
85    </participantTypes>
86    <participants>
87      <participant name="participant1" type="A" selects =" participant2 "/>
88      <participant name="participant2" type="B" selects =" participant3 "/>
```

```
89      < participant  name="participant3"  type="C"/>
90    </ participants >
91    <messageLinks xmlns:msg="http://www.example.org/messages/"> (K1)
92      <messageLink name="Request1Link"
93        sender=" participant1 "  sendActivity ="SendRequest1"
94         receiver =" participant2 "  receiveActivity ="ReceiveRequest1"
95        messageName="Request1" messageType="msg:Request1"/>
96      <messageLink name="DataTransferLink"
97        sender=" participant2 "  sendActivity ="SendData"
98         receiver =" participant1 "  receiveActivity ="ReceiveData"
99        messageName="DataTranser" messageType="msg:DataTransfer"/>
100     <messageLink name="Request2Link"
101       sender=" participant2 "  sendActivity ="SendRequest2"
102        receiver =" participant3 "  receiveActivity ="ReceiveRequest2"
103       messageName="Message2" messageType="msg:Request2"/>
104     <messageLink name="Reply2.1Link"
105       sender=" participant3 "  sendActivity ="SendReply2.1"
106        receiver =" participant2 "  receiveActivity ="ReceiveReply2.1"
107       messageName="Reply2.1" messageType="msg:Reply2.1"/>
108     <messageLink name="Reply2.2Link"
109       sender=" participant3 "  sendActivity ="SendReply2.2"
110        receiver =" participant2 "  receiveActivity ="ReceiveReply2.2"
111       messageName="Reply2.2" messageType="msg:Reply2.2"/>
112     <messageLink name="Reply1Link"
113       sender=" participant2 "  sendActivity ="SendReply1"
114        receiver =" participant1 "  receiveActivity ="ReceiveReply1"
115       messageName="Reply1" messageType="msg:Reply1"/>
116   </messageLinks>
117 </topology>
```

**Listing 4.4:** Umsetzung der Choreographie aus Abschnitt 4.2 in BPEL4Chor

### 4.3.9 Colombo

Colombo ist ein von Berardi et al. [BCD+05] im Jahre 2005 entwickeltes Framework, mit welchem Web Services spezifiziert werden in Form von (1) atomaren Prozessen, welche sie ausführen können; (2) ihre Auswirkungen auf die „reale Welt“, was modelliert ist als relationale Datenbank; (3) ihr Übergangs-basiertes Verhalten; und (4) der Nachrichtenaustausch.

Im Folgenden wird die Eignung von Colombo zur Choreographiemodellierung mittels der in Abschnitt 4.1 vorgestellten Kriterien evaluiert. In Tabelle 4.1, Listing 4.5 und Abbildung 4.12 ist die Choreographie aus Abschnitt 4.2 mittels dem Colombo Framework umgesetzt.

| K1 | K2 | K3 | K4 | K5 | K6 | K7 |
|---|---|---|---|---|---|---|
| ✓ | ✓ | × | ✓ | ✓ | ✓ | ✓ |

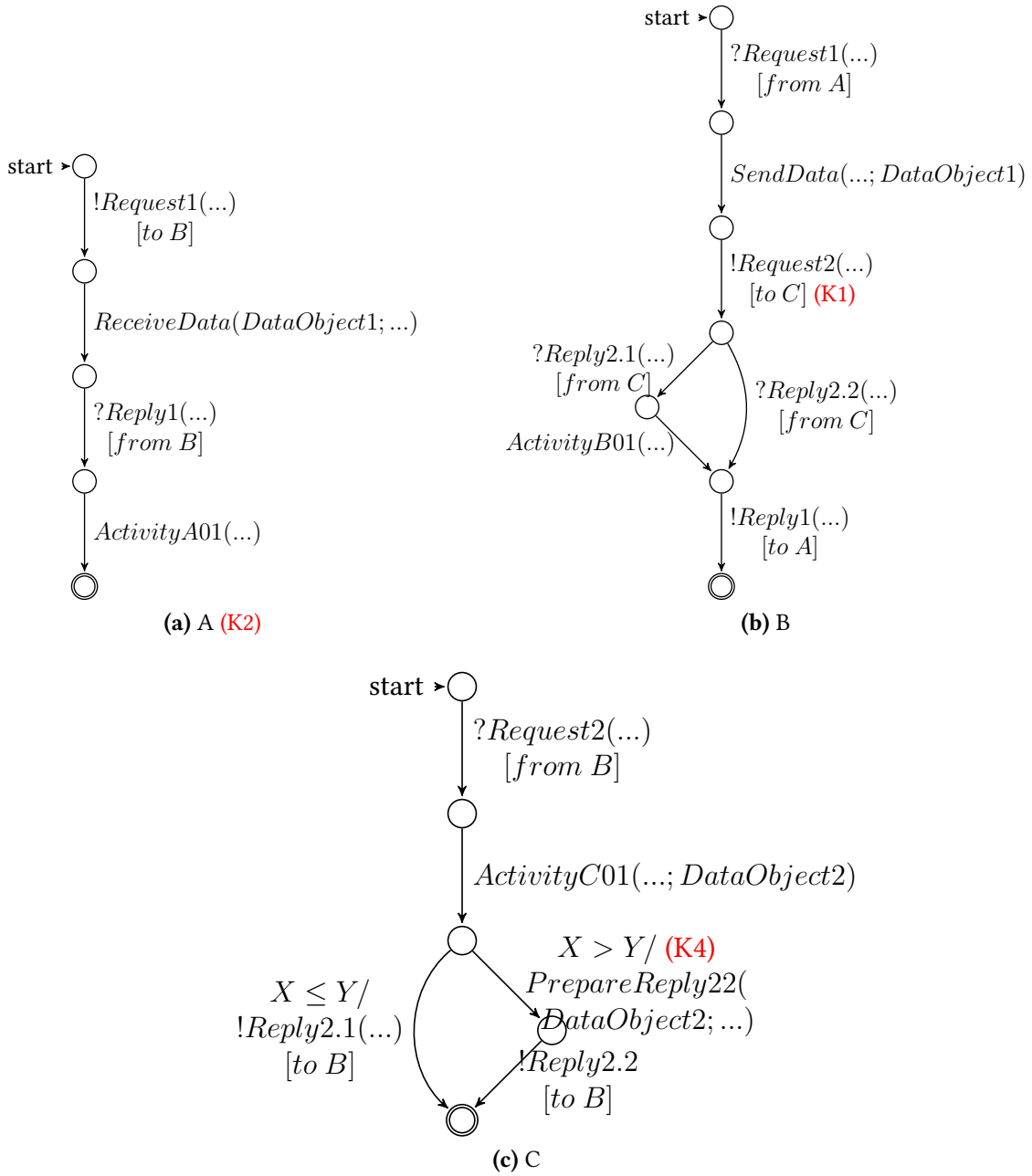
- *(K1) Multilaterale Interaktionen*: Ja, in Colombo lassen sich beliebig viele Teilnehmer spezifizieren, welche beliebig untereinander interagieren können. Bei den Automaten, welche das Übergangs-basierte Verhalten beschreiben, werden eingehende und ausgehende Nachrichten spezifiziert (vgl. Abbildung 4.12), sowie auch interne Aktivitäten angegeben, welche als atomare Prozesse extern genauer definiert werden (vgl. Listing 4.5). Um die einzelnen Interaktionen zwischen den Teilnehmern zu identifizieren, also damit man auch weiß woher eine Nachricht kommt oder an wen eine gesendet wird, werden „Linkages“ definiert. Wir geben in unserem Beispiel aus Abbildung 4.12 diese Linkages in Klammern bei den jeweiligen Nachrichten an. Wie in [BCD+05] verwenden wir in unserem Beispiel ein ! für eine ausgehende Nachricht und ? für eine eingehende Nachricht.
- *(K2) Teilnehmertopologie*: Ja, die einzelnen Teilnehmer werden durch Automaten beim Übergangs-basierten Verhalten dargestellt.
- *(K3) Zeit-Constraints*: Nein, es werden keine Konstrukte zur Modellierung von Zeit-Constraints bereitgestellt.
- *(K4) Daten-basierter Kontrollfluss*: Ja, ein Daten-basierter Kontrollfluss kann modelliert werden. Bei den einzelnen Transitionen lassen sich so bestimmte Datenabfragen modellieren. Nur wenn die Bedingungen erfüllt sind, wird der entsprechende Pfad gelaufen. Innerhalb von atomaren Prozessen lassen sich ebenfalls Daten-basierte Abfragen tätigen.
- *(K5) Modellierung von Datenobjekten*: Ja, im Modell der „realen Welt“ werden die Daten modelliert.
- *(K6) Expliziter Datenfluss*: Ja, die spezifizierten Daten aus dem Modell der realen Welt können als Input und/oder Output bei den einzelnen atomaren Prozesse dienen. Diese Daten können hierbei modifiziert werden. In unserem Beispiel wurde ein neuer atomarer Prozess $PrepareReply22$ hinzugefügt, da ein atomarer Prozess nicht auch direkt einen Nachrichtenaustausch darstellen kann.
- *(K7) Partnerübergreifender Datenfluss*: Ja, die Daten aus der „realen Welt“ dienen als Input und Output bei den Definitionen der atomaren Prozesse. Da die Daten global liegen, gibt es keinen Unterschied zwischen einem expliziten oder partnerübergreifenden Datenfluss. Da atomare Prozesse auf die Daten zugreifen, werden die Aktivitäten $SendData$ und $ReceiveData$ nicht als Nachrichtenaustausch modelliert, sondern als atomare Prozesse, welche dann so modelliert werden, sodass $SendData$ entsprechende Daten in der realen Welt anpasst, welcher als Input für $ReceiveData$ dient.

| | | | | | | |
|---|---|---|---|---|---|---|
| **Data1** | id1 | DataObject1 | b | c | ··· | (K5) |
| | ··· | ··· | ··· | ··· | ··· | |
| **Data2** | id2 | DataObject2 | x | y | ··· | |
| | ··· | ··· | ··· | ··· | ··· | |

**Tabelle 4.1:** Weltschema Instanz in Colombo

```
1  SendData:
2    I: ...
3    O: d1:Dom_=; %Data Object 1    (K6)
4     effects :
5       ...
6
7  ReceiveData:
8    I: d1:Dom_=; %Data Object 1
9    O: ...
10    effects :
11      ...
12
13 ActivityC01:
14   I: ...
15   O: d2:Dom_=; %Data Object 2
16    effects :
17      ...
18
19 PrepareReply22:
20   I: d2:Dom_=; %Data Object 2    (K7)
21   O: ...
22    effects :
23      ...
24
25 ActivityA01: ...
26 ActivityB01: ...
```

**Listing 4.5:** Atomare Prozesse der Choreographie in Colombo




**(a)** A (K2) **(b)** B **(c)** C

**Abbildung 4.12:** Übergangs-basiertes Verhalten der Teilnehmer der Choreographie aus Abschnitt 4.2

### 4.3.10 Interorganizational Workflow Net (IOWF-Net)

Ein „Interorganizational Workflow Net“ (kurz: IOWF-Net) findet seinen Ursprung in Workflow-Netzen ([Van98]), welche eine bestimmte Art von Petri-Netzen sind. IOWF-Nets wurden im Jahr 2001 von van der Aalst und Weske [AW01] definiert, um unternehmensübergreifende Workflows zu modellieren.

Im Folgenden wird die Eignung von IOWF-Nets zur Choreographiemodellierung mittels der in Abschnitt 4.1 vorgestellten Kriterien evaluiert. In Abbildung 4.13 ist eine typische Buchhandel-Choreographie umgesetzt, in welcher ein Kunde ein Buch ausleihen oder kaufen will. Die genaue Spezifikation dieses Beispiels findet sich in [AW01].

| K1 | K2 | K3 | K4 | K5 | K6 | K7 |
|---|---|---|---|---|---|---|
| ✓ | ✓ | ✓ | ✓ | × | × | × |

- *(K1) Multilaterale Interaktionen*: Ja, IOWF-Nets bestehen aus mehreren Workflow-Netzen (kurz: WF-Netz), welche je einen Teilnehmer repräsentieren und Domänen genannt werden, sowie aus verschiedenen Kanälen. Zusätzlich dazu besitzt jede Domäne entsprechende Methoden, welche durch einen Kanalfluss über verschiedene Kanäle miteinander verbunden sind. Durch diese Kanäle und den dazugehörigen Kanalfluss werden die einzelnen Interaktionen zwischen den Teilnehmern modelliert. Eine Beschränkung für die Anzahl der Teilnehmer oder Interaktionen gibt es hierbei nicht. In unserem Beispiel sind es vier Teilnehmer: Kunde ($customer$), Verleger ($publisher$), Buchhandlung ($bookstore$) und Absender ($shipper$). Die Workflow-Netze sind hier wegen der Übersicht nicht dargestellt, sondern lediglich als $N_x^{part}$ gekennzeichnet. In Abbildung 4.14 sieht man eine Übersicht über das dazugehörige Workflow-Netz des Kunden, $N_C^{part}$, und wie dabei das Workflow-Netz mit den Methoden (grau hinterlegte Boxen) zusammenhängt.
- *(K2) Teilnehmertopologie*: Ja, die Teilnehmertopologie entsteht dadurch, dass jeder Teilnehmer durch ein WF-Netz modelliert wird.
- *(K3) Zeit-Constraints*: Ja, in Workflow-Netzen können als zeitliche Beschränkungen modelliert werden, indem eine Timeout-Kondition als Transition modelliert wird.
- *(K4) Daten-basierter Kontrollfluss*: Ja, in den einzelnen Workflow-Netzen können Verzweigungen mit Vorbedingungen sowie Nachbedingungen angegeben werden. Hierbei kann man Daten-abhängige Abfragen tätigen, um den anschließenden Pfad zu wählen. In unserem Beispiel aus [AW01] gibt es keine Daten-basierte Abfrage, weshalb dieses Kriterium in diesem Beispiel nicht visualisiert werden kann.
- *(K5) Modellierung von Datenobjekten*: Nein, es wird keine Modellierung von Daten in IOWF-Net unterstützt.
- *(K6) Expliziter Datenfluss*: Nein, es wird kein expliziter Datenfluss in IOWF-Nets unterstützt.

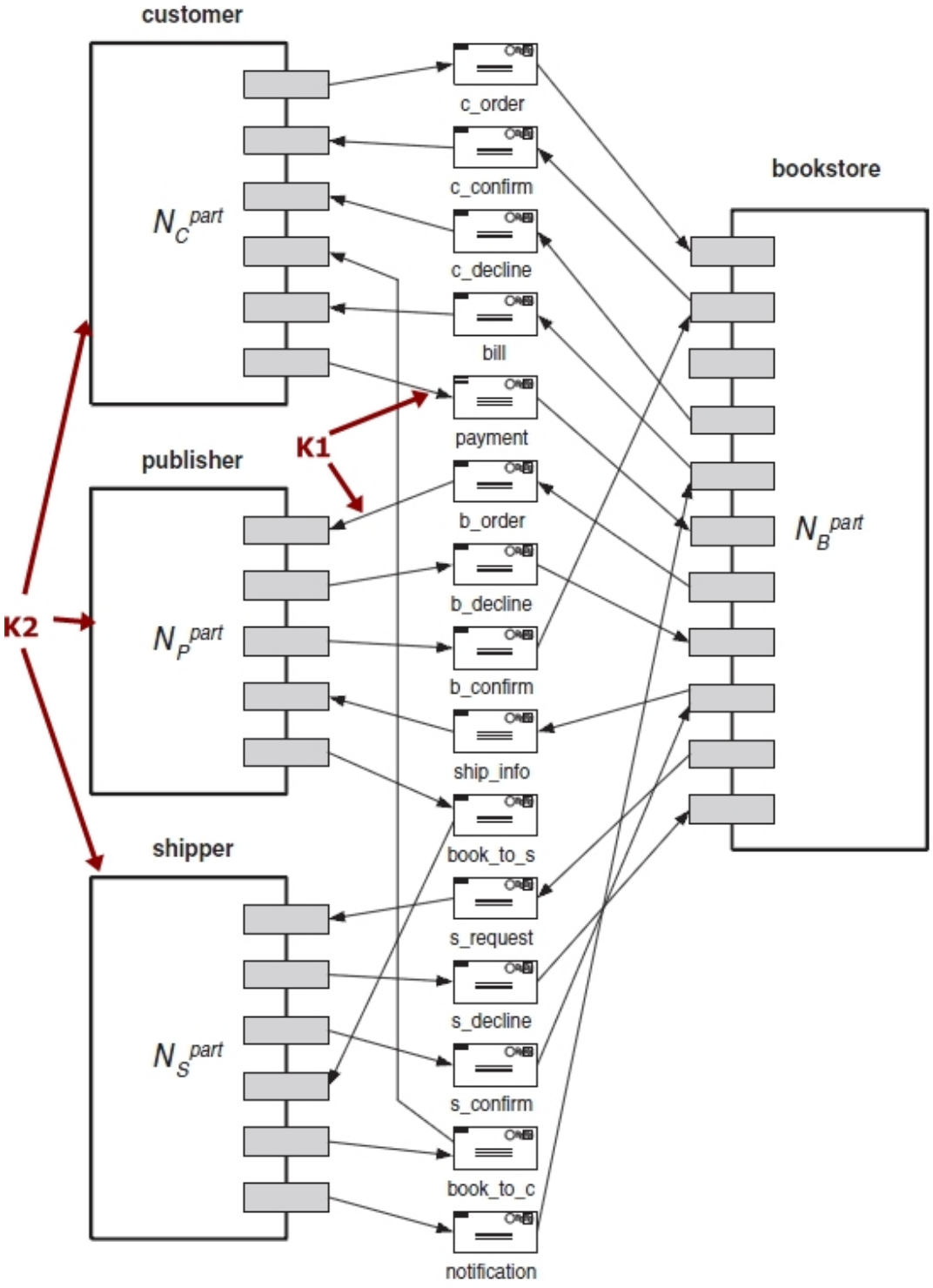



**Abbildung 4.13:** Beispiel einer Choreographie in IOWF-Net aus [AW01]

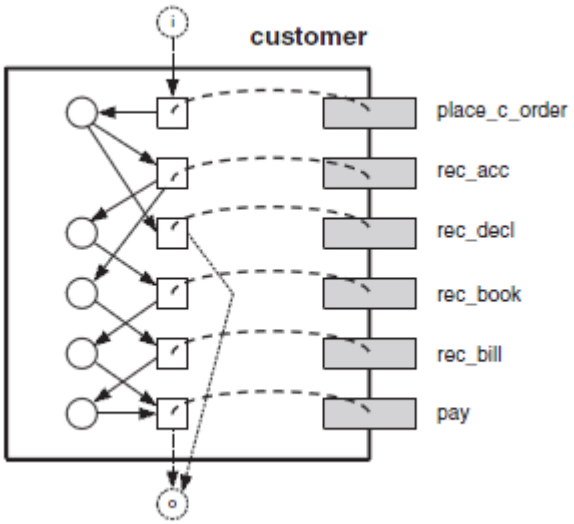



**Abbildung 4.14:** Das Workflow-Netz des Kunden ($N_C^{part}$) aus Abbildung 4.13 [AW01]

- *(K7) Partnerübergreifender Datenfluss*: Nein, es kann kein partnerübergreifender Datenfluss modelliert werden.

### 4.3.11 Colored Petri Nets (CPN)

„Colored Petri Nets“ (kurz: CPN) wurden erstmals im Jahre 1981 von Jensen [Jen81] vorgestellt. Diese CPNs wurden im Laufe der Jahre weiterentwickelt. CPNs erweitern klassische Petri-Netze um Farben zur Modellierung von Daten, Zeit zur Modellierung einer Dauer und Hierarchie (im ursprünglichen Artikel noch nicht definiert) um große Modellierungen zu strukturieren [DLC+07].

Im Folgenden wird die Eignung von CPNs zur Choreographiemodellierung mittels der in Abschnitt 4.1 vorgestellten Kriterien evaluiert. In Abbildung 4.15 ist die Choreographie aus Abschnitt 4.2 mittels einem CPN umgesetzt.

| K1 | K2 | K3 | K4 | K5 | K6 | K7 |
|---|---|---|---|---|---|---|
| ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

- *(K1) Multilaterale Interaktionen*: Ja, in CPNs können beliebig viele Teilnehmer modelliert werden, welche beliebig untereinander kommunizieren können. Diese Teilnehmer werden als sogenannte „Sub-modules“ (auch „Subpages“ genannt) dargestellt in der „Top-Level“ Hierarchieebene. In dieser Ebene werden außerdem die Interaktionen zwischen den Teilnehmern angegeben. Der interne Prozessablauf der einzelnen Teilnehmer wird in der Hierarchieebene darunter via CPNs für jeden Teilnehmer dargestellt. Man kann

auch mehrere Zwischenebenen einbauen. Aus Gründen der Übersichtlichkeit, werden in unserem Beispiel aus Abbildung 4.15 die internen Prozessabläufe der einzelnen Teilnehmer dargestellt, sowie auch die Interaktionen zwischen diesen (in grün), welche eigentlich in der Hierarchieebene darüber genauer spezifiziert werden.

- *(K2) Teilnehmertopologie*: Ja, die Teilnehmertopologie wird durch die einzelnen CPNs der Teilnehmer, auch „sub-modules“ genannt, definiert.
- *(K3) Zeit-Constraints*: Ja, es können Zeit-Constraints modelliert werden, indem bei den Transitionen die zeitliche Bedingungen angegeben werden.
- *(K4) Daten-basierter Kontrollfluss*: Ja, wie Zeit-Constraints, können auch Daten-basierte Bedingungen bei den einzelnen Transitionen angegeben werden.
- *(K5) Modellierung von Datenobjekten*: Ja, Daten werden durch „Colour“ repräsentiert und via „colset“-Konstrukten definiert.
- *(K6) Expliziter Datenfluss*: Ja, jede Aktivität, repräsentiert durch Transitionen, hat einen Input und Output. Somit wird ein expliziter Datenfluss gewährleistet.
- *(K7) Partnerübergreifender Datenfluss*: Ja, es macht für den Datenfluss keinen Unterschied von welchem Service eine Aktivität ausgeführt wird. Somit wird auch ein partnerübergreifender Datenfluss garantiert.

```
1  colset Rq1It =...;
2  colset Data1It =...;
3  colset Rq2It =...;
4  colset DataTOIt =...;
5  colset TimedOut =...;
6  colset C01It =...;
7  colset Rp21It =...;
8  colset Rp22It =...;
9  colset PrepIt =...;
10 colset Rp1It =...;
11 var rq1:Rq1It;
12 var rq2:Rq2It;
13 var c01:C01It;
14 var rp1:Rp1It;
15 var rp21:Rp21It;
16 var rp22:Rp22It;
17 var DObj1:Data1It;
18 var DObj2:Data2It;
19 var dto:DataTOIt;
20 var prep:PrepIt;
21 var failure :TimedOut;
22 var success:Success;
```

**Listing 4.6:** Colorsets und Variablen der Choreographie aus Abschnitt 4.2 definiert für das CPN-Modell aus Abbildung 4.15

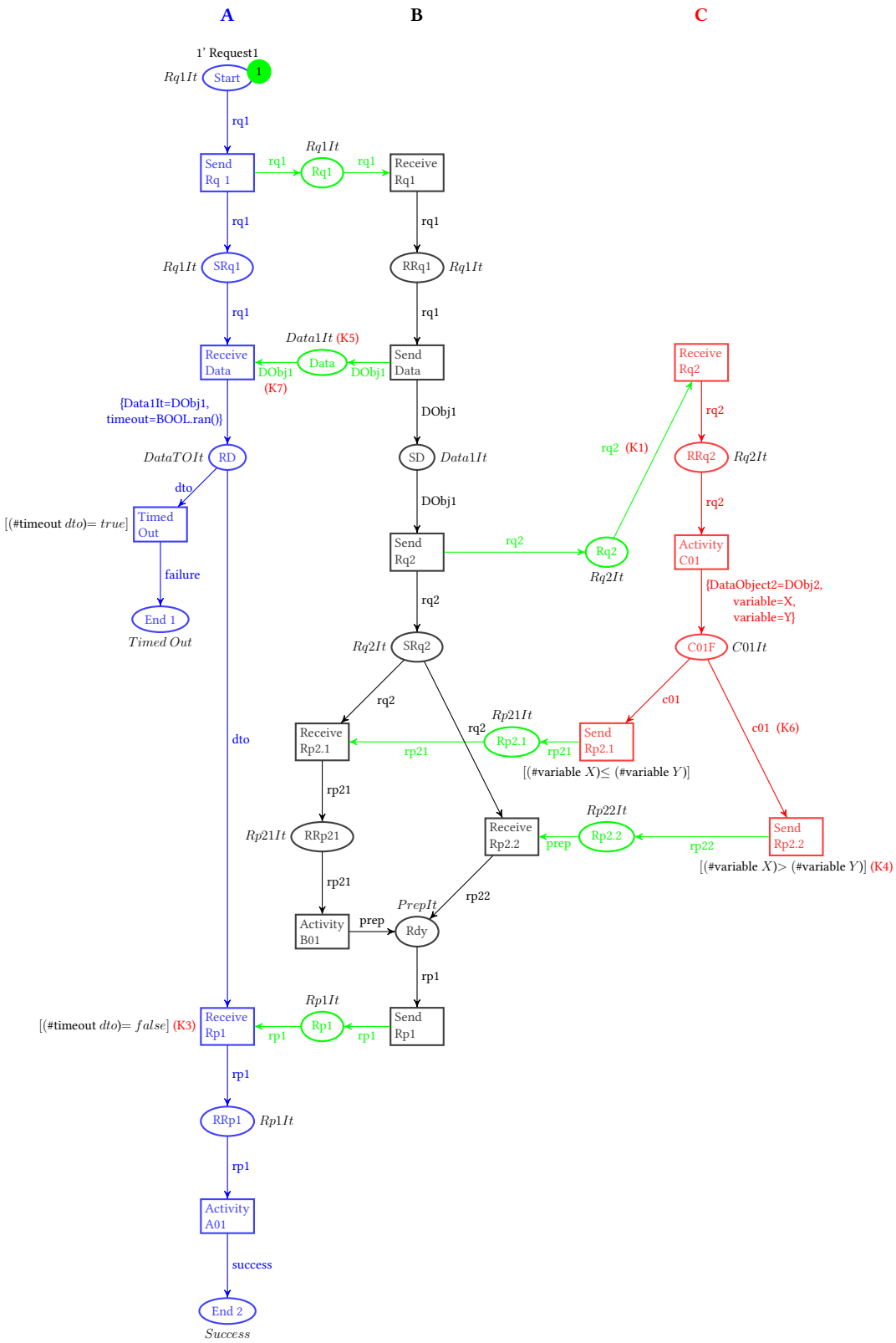



**Abbildung 4.15:** Umsetzung der Choreographie aus Abschnitt 4.2 durch CPN

### 4.3.12 Grid Services Flow Language (GSFL)

Die „Grid Services Flow Language“ (kurz: GSFL) ist eine XML-basierte Sprache zur Modellierung von Workflows von Grid Services, welches aus verschiedenen Modellen besteht: „Activity Model“, „Composition Model“, „Lifecycle Model“, sowie einer zusätzlichen „Service Providers“ Angabe [KWV+02].

Im Folgenden wird die Eignung von GSFL zur Choreographiemodellierung mittels der in Abschnitt 4.1 vorgestellten Kriterien evaluiert.

| K1 | K2 | K3 | K4 | K5 | K6 | K7 |
|---|---|---|---|---|---|---|
| ✓ | ✓ | × | × | ✓ | ✓ | ✓ |

**Hinweis:** Da die einzelnen Konstrukte in der Arbeit von Krishnan, Wagstrom und Von Laszewski nur genannt werden und nicht wie sie genau umgesetzt bzw. modelliert werden, kann ein Beispiel in GSFL nicht gegeben werden. Außerdem können dadurch bei der Analyse teilweise relevante Informationen fehlen.

- *(K1) Multilaterale Interaktionen*: Ja, im Composition Model wird der Kontrollfluss und Datenfluss beschrieben, sowie die Kommunikationen zwischen den einzelnen Services. Ein Begrenzung für die Anzahl der Services, welche die Teilnehmer repräsentieren, gibt es nicht.
- *(K2) Teilnehmertopologie*: Ja, die Teilnehmer werden in der „Service Providers“ Angabe aufgelistet und spezifiziert.
- *(K3) Zeit-Constraints*: Nein, es werden keine Konstrukte zur Modellierung von Zeit-Constraints in GSFL angegeben.
- *(K4) Daten-basierter Kontrollfluss*: Nein, es wird in der Arbeit von Krishnan, Wagstrom und Von Laszewski nicht erwähnt, dass sich der Kontrollfluss durch Daten-basierte Abfragen leiten lässt.
- *(K5) Modellierung von Datenobjekten*: Ja, im „Data Model“, welches ein Teilmodell des „Composition Model“ ist, werden die einzelnen Datenobjekte modelliert.
- *(K6) Expliziter Datenfluss*: Ja, jedes „Data Model“-Element enthält die Attribute „dataInTo“, sowie „dataOutFrom“, womit sich beschreiben lässt, für welche Aktivitäten diese Daten als Input dienen und von welchen Aktivitäten sie der Output sind. Somit wird ein expliziter Datenfluss gewährleistet.
- *(K7) Partnerübergreifender Datenfluss*: Ja, es wird nur erwähnt, dass die Daten direkt als Input oder Ouptut von Aktivitäten modelliert werden können, es wird dabei nicht erwähnt, dass die Aktivitäten nicht von unterschiedlichen Teilnehmern sein dürfen. Somit ist dieses Kriterium wahrscheinlich erfüllt.

### 4.3.13 Let's Dance

„Let's Dance" ist eine von Zaha et al. [ZBDH06] veröffentlichte Choreographiesprache aus dem Jahr 2006. Die Idee der Sprache ist Modelle von Service-Interaktionen von der Verhaltensperspektive aus zu erfassen.

Im Folgenden wird die Eignung von Let's Dance zur Choreographiemodellierung mittels der in Abschnitt 4.1 vorgestellten Kriterien evaluiert. In Abbildung 4.16 ist ein Beispiel einer Kauf-Choreographie aus [DZD06].

| K1 | K2 | K3 | K4 | K5 | K6 | K7 |
|---|---|---|---|---|---|---|
| ✓ | × | ✓ | ✓ | × | × | × |

- *(K1) Multilaterale Interaktionen*: Ja, in Let's Dance werden die Interaktionen zwischen den verschiedenen Teilnehmern modelliert. Dabei werden bei jeder Interaktion die Rollen, sowie die dafür zuständigen Teilnehmer, angegeben. Dabei gibt es für jede Rolle einen Akteur, der diese Rolle ausführt. Es gibt dabei keine Beschränkungen für die Anzahl der Rollen und Akteure in einer Let's Dance Choreographie, weshalb das Kriterium „Multilaterale Interaktionen" erfüllt ist. In unserem Beispiel gibt es nur zwei Rollen: Käufer ($Buyer$) und Lieferant ($Supplier$), welche von den Teilnehmern $b1$ und $s1$ ausgeführt werden. Die erste Interaktion $Order$ wird vom Käufer initiiert, wodurch die Choreographie instanziiert wird.
- *(K2) Teilnehmertopologie*: Nein, die Teilnehmer werden nur implizit über die einzelnen Interaktionen angegeben. Eine Übersicht über alle Teilnehmer gibt es somit nicht.
- *(K3) Zeit-Constraints*: Ja, „Timer" können in Let's Dance als Subtyp von Interaktionen angegeben werden, mithilfe derer sich zeitliche Beschränkungen modellieren lassen. Im Beispiel aus Abbildung 4.16 ist kein Beispiel davon angegeben, sowie auch nicht in den restlichen Beispielen der jeweiligen Artikeln ([DKZD06; DZD06; ZBDH06]).
- *(K4) Daten-basierter Kontrollfluss*: Ja, die Ausführung der verschiedenen Interaktionen kann an bestimmte Daten-basierte Konditionen gebunden sein. Im Beispiel aus Abbildung 4.16 sind Konditionen angegeben, z.B. $able\ to\ cancel\ (s1)$, welche allerdings nicht Daten-basiert sind. Daten-basierte Konditionen werden aber ebenfalls so angegeben, z.B. $if\ X > Y\ (s1)$. Der Teilnehmer welcher für die Abfrage zuständig ist, wird in Klammern angegeben. In diesem Fall ist es der Teilnehmer $(s1)$.
- *(K5) Modellierung von Datenobjekten*: Nein, in Let's Dance wird lediglich der Nachrichtenaustausch und die Interaktionen zwischen den Teilnehmern modelliert. Es unterstützt es keine Modellierung von Datenobjekten.
- *(K6) Expliziter Datenfluss*: Nein, ein expliziter Datenfluss kann in Let's Dance nicht modelliert werden.
- *(K7) Partnerübergreifender Datenfluss*: Nein, ein partnerübergreifender Datenfluss kann in Let's Dance ebenfalls nicht modelliert werden.

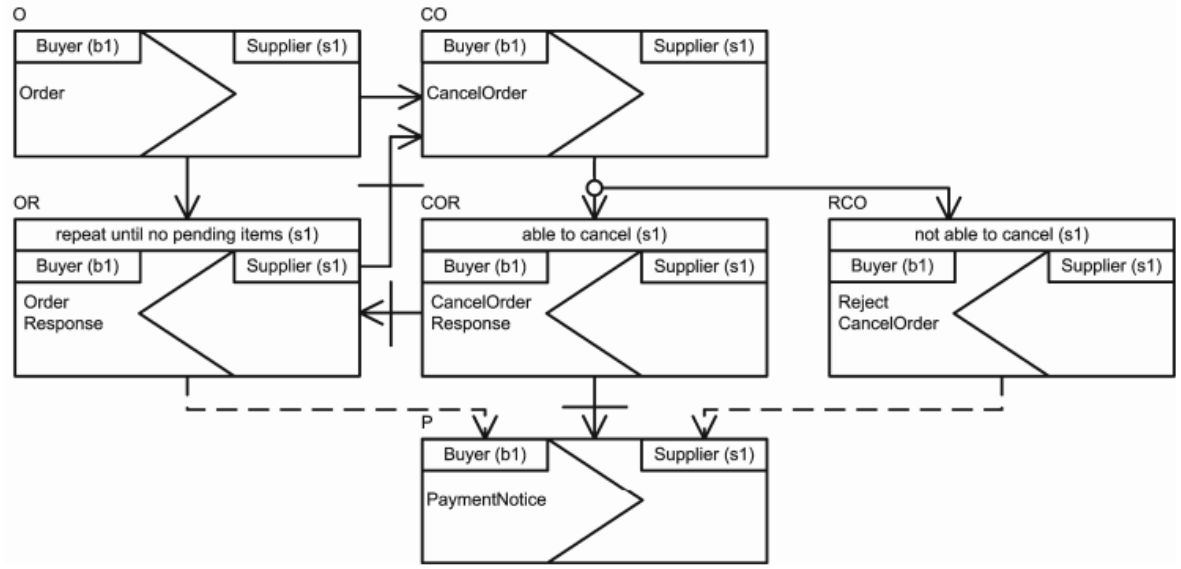



**Abbildung 4.16:** Beispiel einer Choreographie in Let's Dance aus [DZD06]

### 4.3.14 Open Workflow Nets (oWFN)

Van der Aalst hat bereits in [Van98] eine spezielle Klasse von Petri Netzen, sogenannte „Workflow Nets" (kurz: WFNs), spezifiziert, um die Struktur von Workflows zu beschreiben. Da jedoch auch Workflow Services miteinander kommunizieren können, müssen weitere Konstrukte her, um die Kommunikationen zu modellieren. „Open Workflow Nets" (kurz: oWFN) bieten solche Konstrukte [MRS05].

Im Folgenden wird die Eignung von oWFNs zur Choreographiemodellierung mittels der in Abschnitt 4.1 vorgestellten Kriterien evaluiert. In Abbildung 4.17 ist die Choreographie aus Abschnitt 4.2 mittels einem zusammengesetzten oWFN umgesetzt.

| K1 | K2 | K3 | K4 | K5 | K6 | K7 |
|---|---|---|---|---|---|---|
| ✓ | ✓ | ✓ | ✓ | × | × | × |

- *(K1) Multilaterale Interaktionen*: Ja, ein Prozess eines Teilnehmers wird durch ein oWFN repräsentiert, welches eine Erweiterung eines Workflow-Netzes ist, erweitert durch sogenannten „Communication Places", welche jeweils einen Channel zum Senden und Empfangen von Nachrichten repräsentieren. Durch die Komposition der einzelnen oWFNs ergibt sich die Choreographie. Zur übersichtlichen Modellierung, verwenden wird die Schreibweise aus [MRS05] in unserem Beispiel aus Abbildung 4.17 und schreiben somit für jede ausgehende Nachricht ein ! und für eingehende Nachricht ein ? davor.

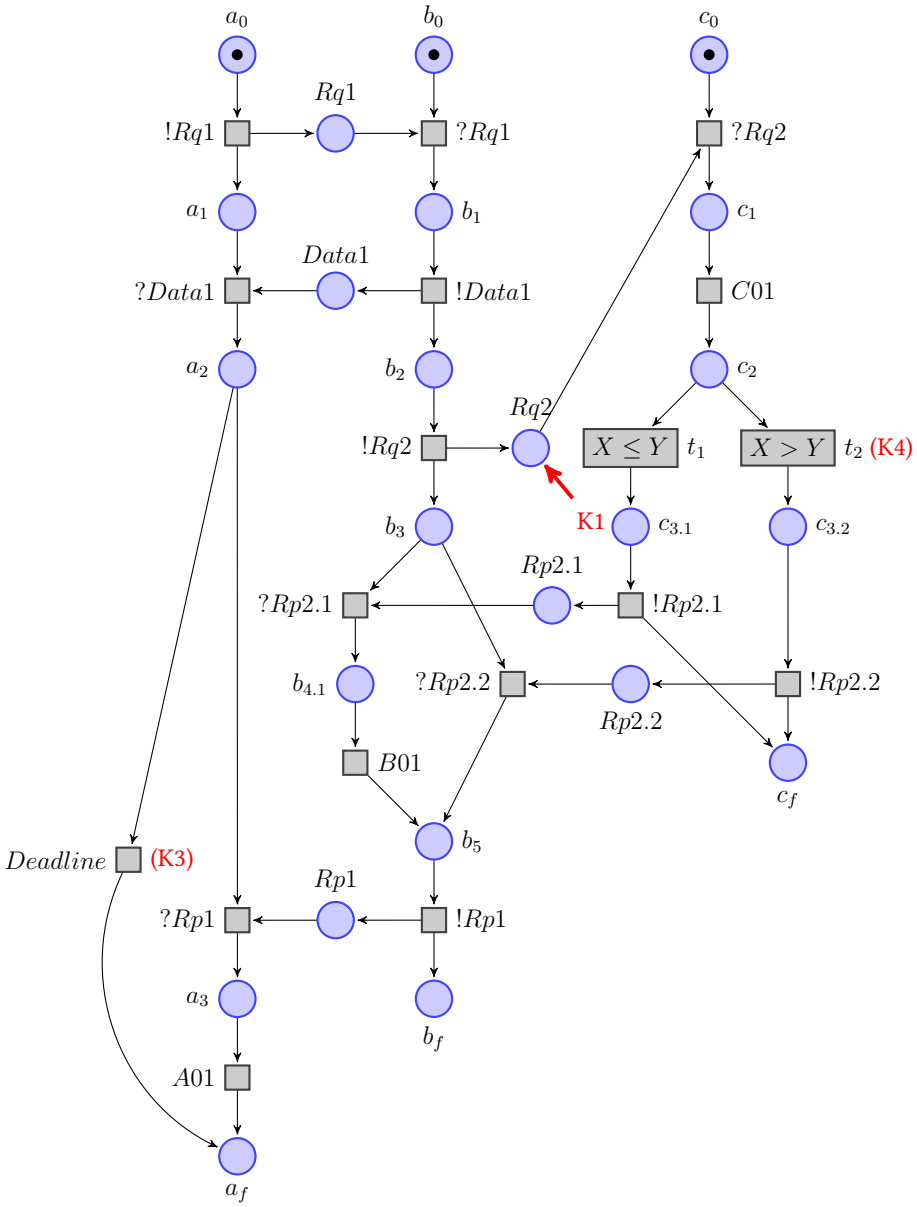



**Abbildung 4.17:** Umsetzung der Choreographie aus Abschnitt 4.2 durch die Komposition der oWFNs der einzelnen Teilnehmer: $P_{A \oplus B \oplus C}$

- *(K2) Teilnehmertopologie*: Ja, die Teilnehmertopologie ergibt sich durch die verschiedenen oWFNs der einzelnen Teilnehmer. Die einzelnen oWFNs der Teilnehmer haben wir in unserem Beispiel nicht realisiert.
- *(K3) Zeit-Constraints*: Ja, Zeit-Constraints können via Transitionen angegeben werden.
- *(K4) Daten-basierter Kontrollfluss*: Ja, es können Abzweigungen modelliert werden, wobei die einzelnen Pfade von Daten-basierten Abfragen abhängen.
- *(K5) Modellierung von Datenobjekten*: Nein, die Modellierung von Datenobjekten ist in oWFN nicht möglich.
- *(K6) Expliziter Datenfluss*: Nein, ein expliziter Datenfluss kann in einem oWFN nicht modelliert werden.
- *(K7) Partnerübergreifender Datenfluss*: Nein, in oWFNs wird lediglich der Nachrichtenaustausch modelliert. Deshalb kann ein partnerübergreifender Datenfluss in oWFNs nicht gewährleistet werden.

### 4.3.15 Chor

„Chor“ ist eine von Qiu et al. [QZCY07]definierte formale Sprache zur Modellierung von Choreographien. Es gilt als ein vereinfachtes formales Modell von WS-CDL.

Im Folgenden wird die Eignung von Chor zur Choreographiemodellierung mittels der in Abschnitt 4.1 vorgestellten Kriterien evaluiert. In Abbildung 4.18 ist die Choreographie aus Abschnitt 4.2 mittels Chor umgesetzt.

| K1 | K2 | K3 | K4 | K5 | K6 | K7 |
|---|---|---|---|---|---|---|
| ✓ | ✓ | × | × | × | × | × |

- *(K1) Multilaterale Interaktionen*: Ja, in Chor lassen sich beliebig vielen Rollen modellieren. Die Kommunikationen zwischen den Rollen ist unbeschränkt möglich. Somit werden multilaterale Interaktionen unterstützt. Da Chor eine formale Sprache ist, werden hier alle Rollen in einer Menge angegeben, so wie ihre zugehörigen Aktivitäten. In unserem Beispiel sind das $R_{C_1}$ für die Rollen und $locals(C_1)$ für die Aktivitäten, wobei die einzelnen Aktivitäten in der Form $Name\_der\_Aktivität^{Teilnehmer\_der\_sie\_ausführt}$ angegeben sind. Eine Kommunikation geschieht in der Form $Name\_der\_Interaktion^{[Teilnehmer1,Teilnehmer2]}$. Der Inhalt der Nachrichten und die ausgetauschten Daten werden in Chor nicht berücksichtigt.
- *(K2) Teilnehmertopologie*: Ja, alle Rollen werden in einer Choreographie zusammen in einer Menge aufgelistet.
- *(K3) Zeit-Constraints*: Nein, die Modellierung von zeitlichen Beschränkungen wird in Chor nicht unterstützt.

$$R_{C_1} = \{A, B, C\} \quad \text{(K2)}$$
$$locals(C_1) = \{ActivityA01^A, ActivityB01^B, ActivityC01^C\}$$

$$C_1 = Request1^{[A,B]}; DataTransfer^{[B,A]}; Request2^{[B,C]}; ActivityC01^C; ((Reply2.1^{[C,B]}; ActivityB01^B) \sqcap Reply2.2^{[C,B]}); Reply1^{[B,A]}; ActivityA01^A \quad \text{(K1)}$$

**Abbildung 4.18:** Umsetzung der Choreographie aus Abschnitt 4.2 in Chor

- *(K4) Daten-basierte Abfragen*: Nein, in Chor gibt es zwar ein „Choice"-Element, mithilfe dessen man wählen kann, welche Aktion als nächstes ausgeführt wird oder welchen Pfad man weiterläuft, aber es wird nicht spezifiziert, dass man die Wahl durch Daten-basierte Abfragen treffen kann. Es wird nur erwähnt, dass eine Rolle die Entscheidung treffen muss. Dieses Choice-Element wird durch das Symbol $\sqcap$ repräsentiert.
- *(K5) Modellierung von Datenobjekten*: Nein, in Chor gibt es keine explizite Modellierung von Datenobjekten.
- *(K6) Expliziter Datenfluss*: Nein, ein expliziter Datenfluss kann in Chor nicht modelliert werden.
- *(K7) Partnerübergreifender Datenfluss*: Nein, ein partnerübergreifender Datenfluss kann in Chor ebenfalls nicht realisiert werden.

### 4.3.16 Deterministic Finite State Automata (DFA)

Ein „Deterministic Finite State Automata" (kurz: DFA, auf deutsch: „Deterministischer endlicher Automat" oder kurz: DEA) wird im Artikel von Wombacher et al. [WFMN04] zur Modellierung von Choreographien verwendet. Jedoch hat dieser deterministische endliche Automat in der Definition aus diesem Artikel eine zusätzliche Bedingung, welche es für die ursprünglich definierten deterministische endliche Automaten nicht gibt. Diese Bedingung ist, dass die Zustandsübergänge eine bestimmte Form einhalten müssen: $from\#to\#message\_name$.

Im Folgenden wird die Eignung von DFA's zur Choreographiemodellierung mittels der in Abschnitt 4.1 vorgestellten Kriterien evaluiert. In Abbildung 4.19 ist die Choreographie aus Abschnitt 4.2 mittels einem DFA umgesetzt.

| K1 | K2 | K3 | K4 | K5 | K6 | K7 |
|---|---|---|---|---|---|---|
| ✓ | × | × | × | × | × | × |

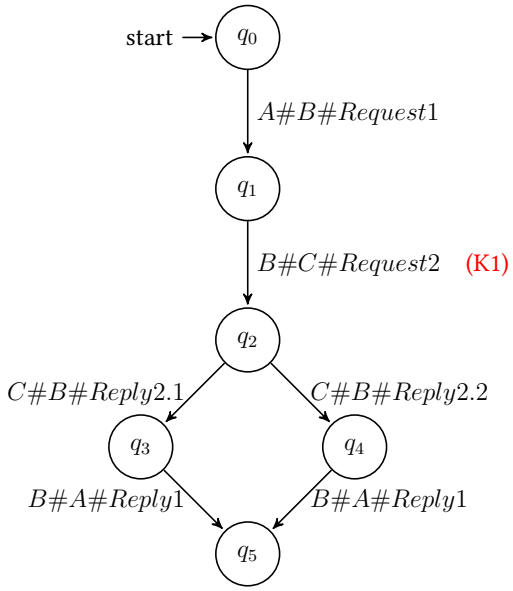



**Abbildung 4.19:** Umsetzung der Choreographie aus Abschnitt 4.2 mit einem DFA

- *(K1) Multilaterale Interaktionen*: Ja, in einer Choreographie in einem DFA kann es beliebig viele Teilnehmer geben, welche untereinander interagieren. Ein DFA unterscheidet sich, wie in der Einleitung der Sprache erwähnt, in einem Punkten von einem „normalen“ deterministischen endlichen Automaten: Die Zustandsübergänge beschreiben die Interaktionen zwischen zwei Teilnehmern in der Form $from\#to\#message\_name$. Diese Form muss eingehalten werden.
- *(K2) Teilnehmertopologie*: Nein, die Teilnehmer werden implizit bei den Interaktionen in den Transitionen angegeben.
- *(K3) Zeit-Constraints*: Nein, in einem DFA gibt es keine Konstrukte zur Modellierung von Zeit-Constraints.
- *(K4) Daten-basierter Kontrollfluss*: Nein, Daten-bedingte Konditionen können in DFAs nicht angegeben werden. In dem Beispiel aus Abbildung 4.19 wird zwar von Zustand $q_2$ eine Abzweigung modelliert, aber die Wahl des Pfades kann nicht Daten-basiert gefällt werden.
- *(K5) Modellierung von Datenobjekten*: Nein, es gibt keine Konstrukte zur Modellierung von Datenobjekten in DFAs.
- *(K6) Expliziter Datenfluss*: Nein, einen expliziten Datenfluss gibt es nicht.
- *(K7) Expliziter und Partnerübergreifender Datenfluss*: Nein, auch ein partnerübergreifender Datenfluss existiert in DFA’s nicht.

$$V_A = \{someData, data_{1a}, request1Data_A, reply1Data_A\} \quad \text{(K5)}$$
$$V_B = \{DataObject1, request1Data_B, request2Data_B, reply2Data_B, reply1Data_B\}$$
$$V_C = \{X_C, Y_C, DataObject2, request2Data_C, reply21Data_C, reply22Data_C\}$$

$$Role = \{(A, \{(Request1, ow)\}, V_A), (B, \{(Request2, ow), (Reply1, ow)\}, V_B),$$
$$(C, \{(Reply2.1, ow), (Reply2.2, ow)\}, V_C)\} \quad \text{(K2)}$$

$C_1$ = (A, B, Request 1, $request1Data_A$, $request1Data_B$, $\uparrow$); (B, A, Data Transfer, $DataObject1$, $data_{1a}$, $\downarrow$); (B, C, Request 2, $request2Data_B$, $request2Data_C$, $\uparrow$); ($X_C \leq Y_C$?(C, B, Reply 2.1, $reply21Data_C$, $reply2Data_B$, $\downarrow$) $\oplus$ $X_C > Y_C$?(C, B, Reply 2.2, $reply22Data_C$, $reply2Data_B$, $\downarrow$)); (B, A, Reply 1, $reply1Data_B$, $reply1Data_A$, $\downarrow$) (K1), (K4)

$$X = someData = data_{1a} = request1Data_A = reply1Data_A = DataObject1 =$$
$$request1Data_B = request2Data_B = reply2Data_B = reply1Data_B = X_C = Y_C =$$
$$DataObject2 = request2Data_C = reply21Data_C = reply22Data_C = \bot$$

$$\mathcal{C} = (C_1, Role, X)$$

**Abbildung 4.20:** Umsetzung der Choreographie aus Abschnitt 4.2 mit „Bologna“

### 4.3.17 “Bologna“

„Bologna“ ist ein von Busi et al. [BGG+05b] im Jahre 2005 definiertes formales Modell zur Beschreibung von Choreographien, welches durch WS-CDL inspiriert wurde. Die Autoren haben dabei keinen Namen für diese Choreographiesprache angegeben. In [SBFZ08] wird sie unter dem Namen „Bologna“ referenziert, da sie an der Universität von Bologna entwickelt wurde, weshalb wir diesen Namen hier ebenfalls verwenden.

Im Folgenden wird die Eignung von „Bologna“ zur Choreographiemodellierung mittels der in Abschnitt 4.1 vorgestellten Kriterien evaluiert. In Abbildung 4.20 ist die Choreographie aus Abschnitt 4.2 mittels „Bologna“ umgesetzt.

| K1 | K2 | K3 | K4 | K5 | K6 | K7 |
|---|---|---|---|---|---|---|
| ✓ | ✓ | × | ✓ | ✓ | × | × |

- *(K1) Multilaterale Interaktionen*: Ja, mit „Bologna“ lassen sich beliebig viele Interaktionen zwischen beliebig vielen Teilnehmern modellieren. Eine Choreographie $\mathcal{C}$ besteht aus einem Tupel $(C, \Sigma, X)$, wobei $C \in CL_P$ und $CL_P$ sich über die gegebenen Konversationen (also allen verschiedenen $C$'s) erstreckt, $\Sigma \subset Role$ eine endliche Menge von

allen involvierten Rollen und $X$ eine Kondition ist, welches die Bedingungen des Startzustands beschreibt. In $C$ werden die einzelnen Konversationen und die Reihenfolge dieser Konversationen deklariert. Eine Konversation $m$ besteht aus einem Tupel der Form $\eta ::= (\rho_A, \rho_B, o, \tilde{x}, \tilde{y}, dir)$. $\rho_A$ ist der Initiator der Nachricht, $\rho_B$ der Empfänger, $o$ ist der Name der Operation, $\tilde{x}$ und $\tilde{y}$ die Variablen welche vom Sender und Empfänger benutzt werden. $dir$ ($\in \{\uparrow, \downarrow\}$) gibt an, ob die Konversation eine Anfrage ($\uparrow$) oder eine Antwort ($\downarrow$) ist. Wir haben nur ein $C$ definiert, welches die komplette Konversationsabfolge der Beispielchoreographie beschreibt, weshalb wir $C = C_1$ wählen. Wir wählen $X = someData = data_{1a} = request1Data_A = reply1Data_A = DataObject1 = request1Data_B = request2Data_B = reply2Data_B = reply1Data_B = X_C = Y_C = DataObject2 = request2Data_C = reply21Data_C = reply22Data_C = \perp$, wobei $\perp$ bedeutet dass eine Variable noch nicht initialisiert ist.

- *(K2) Teilnehmertopologie*: Ja, die Teilnehmer werden zusätzlich in einer Menge aufgefasst in der Form $\{(\rho, \omega, V) | \rho \in RName, \omega \subset Op, V \subset Var\}$, wobei $RName$ eine Liste der Rollennamen ist und $Var$ eine Liste von Variablen. $Op$ wird hierbei definiert als $\{(o, t) | o \in OpName, t \in OpType\}$, wobei $OpName$ eine Liste der Operationennamen ist und $OpType$ der Typ der Operation, welcher entweder „One-Way“ ($ow$) oder „Request-Response“ ($rr$) annehmen kann. Da in unserem Beispiel nur die Rollen definiert worden sind, welche an der Choreographie teilnehmen, ist in unserem Fall $\Sigma = Role$.
- *(K3) Zeit-Constraints*: Nein, Konstrukte zur Modellierung von Zeit-Constraints werden in „Bologna“ nicht gegeben.
- *(K4) Daten-basierter Kontrollfluss*: Ja, in „Bologna“ kann man die Erfüllung einer Kondition abfragen, wodurch man entscheiden kann ob ein bestimmter Pfad genommen wird oder nicht. Die abgefragten Konditionen $\mathcal{X}$ werden gefolgt von einem Fragezeichen und den entsprechenden Konversationen, welche durchgeführt werden, sofern die Kondition erfüllt ist. In diesem Fall sind die Abfragen $X_C > Y_C$ bzw. $X_C \leq Y_C$. Durch das exklusive Oder $\oplus$ wird außerdem gewährleistet, dass immer nur einer der beiden Pfade ausgeführt wird.
- *(K5) Modellierung von Datenobjekten*: Ja, die relevanten Daten werden als Variablen in einer Menge definiert. In unserem Beispiel sind die Mengen $V_A$, $V_B$ und $V_C$ für die jeweiligen Teilnehmer.
- *(K6) Expliziter Datenfluss*: Nein, einen expliziten Datenfluss gibt es nicht.
- *(K7) Partnerübergreifender Datenfluss*: Nein, ein partnerübergreifender Datenfluss kann in „Bologna“ nicht gewährleistet werden. Der Austausch der Daten muss über Nachrichten geschehen.

### 4.3.18 Multiagent Protocols (MAP)

Der „Multiagent Protocols“ (kurz: MAP) ist ein von Barker, Walton und Robertson [BWR09] definierte formale Syntax zur Beschreibung von Choreographien, welche speziell für Choreographien definiert wurde.

Im Folgenden wird die Eignung von MAP zur Choreographiemodellierung mittels der in Abschnitt 4.1 vorgestellten Kriterien evaluiert. In Listing 4.7 ist die Choreographie aus Abschnitt 4.2 mittels MAP umgesetzt.

| K1 | K2 | K3 | K4 | K5 | K6 | K7 |
|---|---|---|---|---|---|---|
| ✓ | ✓ | ✓ | (✓) | × | × | × |

- *(K1) Multilaterale Interaktionen*: Ja, es lassen sich mehrere Teilnehmer modellieren, welche beliebig untereinander interagieren können. In MAP unterscheidet man zwischen sogenannten „Peers“ und „Rollen“. Jeder Peer hat einen eigenen Namen und muss einer Rolle zugeordnet werden. Dabei können mehrere Peers der gleichen Rolle angehören. Im in Listing 4.7 dargestellten Beispiel bedeutet $request(\$Request1, \$DataObject1) => peer(\_, \%B)$ hierbei, dass eine Nachricht des Typs $request$ mit dem Parameter $\$Request1$ an alle Peers mit der Rolle $\%B$ gesendet wird. Wäre der Pfeil $<=$, dann würde es bedeuten dass man von allen Peers der Rolle $\%B$ solch eine Nachricht erhält. In diesem Beispiel geben wir die Informationen einer Nachricht als Parameter weiter, so wie auch die entsprechenden Daten, falls welche ausgetauscht werden.
- *(K2) Teilnehmertopologie*: Ja, eine Teilnehmertopologie ergibt sich durch die Auflistung der Rollen im „Protokoll“.
- *(K3) Zeit-Constraints*: Ja, es lassen sich Zeit-Constraints über das Konstrukt „Timeout“ definieren. Timeouts können in MAP nur als bestimmte Dauer angegeben werden (in Sekunden), für wie lange man auf eine Antwort wartet. Ein Datum kann man nicht direkt angeben. Deshalb wurde in unserem Beispiel die Zahl $5000$ gewählt, was bedeutet dass $\%A$ nach dem Erhalten des Datenobjekts 1, $5000$ Sekunden lang auf eine Antwort von $\%B$ wartet, bevor die Operationen im Timeout-Block ausgelöst werden.
- *(K4) Daten-basierter Kontrollfluss*: Teilweise, MAP erlaubt nicht direkt einen Daten-basierten Kontrollfluss. Es wird aber erwähnt, dass es möglich ist hierfür eine Lösung zu implementieren, wodurch das Kriterium teilweise erfüllt ist. Der $or\ else$-Block in unserem Beispiel wird in MAP allerdings nur ausgeführt, wenn der Block davor eine Aktion enthält, welche fehlgeschlagen ist.
- *(K5) Modellierung von Datenobjekten*: Nein, es gibt keine Konstrukte zur expliziten Modellierung von Datenobjekten in MAP. Die Daten werden implizit über die Interaktionen mit angegeben.
- *(K6) Expliziter Datenfluss*: Nein, einen expliziten Datenfluss gibt es nicht.

- *(K7) Partnerübergreifender Datenfluss*: Nein, ein partnerübergreifender Datenfluss lässt sich in MAP nicht realisieren. Der Datenaustausch geschieht über Nachrichten.

```
1  %A{    (K2)
2  method main() =
3    request($Request1) => peer(_, %B)
4    then
5      reply($SendData, $DataObject1) <= peer(_, %B))
6    then waitfor
7      (reply($Reply1) <= peer(_, %B))
8    timeout[5000](e)     (K3)
9  }
10
11 %B{
12 method main() =
13   request($Request1, $DataObject1) <= peer(_, %A)
14   then
15     request($Request2) => peer(_, %C)    (K1)
16   then
17     reply($Reply2.1) <= peer(_, %C)
18       or else reply($Reply2.2) <= peer( _, %C)
19   then
20     reply($Reply1) => peer(_, %A)
21 }
22
23 %C{
24 method main() =
25   request($Request2) <= peer(_, %B)
26   then
27     reply($Reply2.1) => peer(_, %B)
28       or else reply($Reply2.2) => peer(_, %B)
29 }
```

**Listing 4.7:** Umsetzung der Choreographie aus Abschnitt 4.2 mit MAP

## 4.4 Zusammenfassung der Analyse

In Tabelle 4.2 ist die Analyse der einzelnen Sprachen in einer übersichtlichen Tabelle zusammengefasst, damit man einen direkten Vergleich über das Abschneiden der einzelnen Sprachen hat. Diese Tabelle ist sortiert nach der Anzahl der zutreffenden Kriterien für eine Sprache. Bei gleich vielen zutreffenden Kriterien, werden die Sprachen lexikografisch in der Tabelle angeordnet. Teilweise erfüllte Kriterien werden ebenfalls mit einbezogen.

Wie man aus der Tabelle entnehmen kann, haben lediglich UML Aktivitätsdiagramme und „Colored Petri Nets“ (CPN) alle Kriterien erfüllt. Dies bedeutet allerdings nicht, dass diese Sprachen zwingend am besten geeignet sind zur Modellierung von Choreographien, da es noch einige zusätzliche Kriterien gibt, welche in dieser Arbeit nicht untersucht wurden.

In Abbildung 4.21 sieht man von wieviel Prozent der Sprachen die einzelnen Kriterien erfüllt werden. Es ist dabei keine Überraschung, dass das erste Kriterium von allen Sprachen erfüllt wird, da es auch als notwendige Bedingung definiert wurde, welche Choreographiesprachen erfüllen müssen. Etwas überraschend ist hingegen, dass sich zwar in $81.3\%$ der Sprachen ein Daten-basierter Kontrollfluss modellieren lassen, aber sich nur in $45.8\%$ der Sprachen Datenobjekte explizit modellieren lassen. Die Daten werden oft nur implizit modelliert oder auch gar nicht, obwohl Abfragen getätigt werden können, welche sich auf bestimmte Daten beziehen. Außerdem zeigt diese Analyse, dass viele Choreographiesprachen keine expliziten oder partnerübergreifende Datenflüsse unterstützen. In den untersuchten Sprachen lassen sich die Daten oft nur über Nachrichten austauschen.

| | (K1) Multilaterale Interaktionen | (K2) Teilnehmertopologie | (K3) Zeit-Constraints | (K4) Daten-basierter Kontrollfluss | (K5) Modellierung von Datenobjekten | (K6) Expliziter Datenfluss | (K7) Partnerübergreifender Datenfluss |
|---|---|---|---|---|---|---|---|
| CPN | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| UML Aktivitätsdiagramm | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| BPMN 2.0 Kollaborationsdiagramm | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | × |
| BPMN 2.0 Konversationsdiagramm | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | × |
| BPEL4Chor | ✓ | ✓ | ✓ | ✓ | ✓ | × | × |
| Colombo | ✓ | ✓ | × | ✓ | ✓ | ✓ | ✓ |
| EPC | ✓ | × | × | ✓ | ✓ | ✓ | ✓ |
| GSFL | ✓ | ✓ | × | × | ✓ | ✓ | ✓ |
| WS-CDL | ✓ | ✓ | ✓ | ✓ | ✓ | × | × |
| WSFL | ✓ | ✓ | × | ✓ | ✓ | ✓ | × |
| „Bologna“ | ✓ | ✓ | × | ✓ | ✓ | × | × |
| DecSerFlow | ✓ | ✓ | ✓ | ✓ | × | × | × |
| IOWF-Net | ✓ | ✓ | ✓ | ✓ | × | × | × |
| oWFN | ✓ | ✓ | ✓ | ✓ | × | × | × |
| UML Sequenzdiagramm | ✓ | ✓ | ✓ | ✓ | × | × | × |
| MAP | ✓ | ✓ | ✓ | (✓) | × | × | × |
| BPMN 2.0 Choreographiediagramm | ✓ | × | ✓ | ✓ | × | × | × |
| Let’s Dance | ✓ | × | ✓ | ✓ | × | × | × |
| UML Interaktionsübersichtsdiagramm | ✓ | × | ✓ | ✓ | × | × | × |
| UML Kommunikationsdiagramm | ✓ | ✓ | × | ✓ | × | × | × |
| UML Zeitdiagramm | ✓ | ✓ | ✓ | × | × | × | × |
| Global Calculus | ✓ | × | (✓) | ✓ | × | × | × |
| Chor | ✓ | ✓ | × | × | × | × | × |
| DFA | ✓ | × | × | × | × | × | × |

**Tabelle 4.2:** Vergleich der Choreographiesprachen

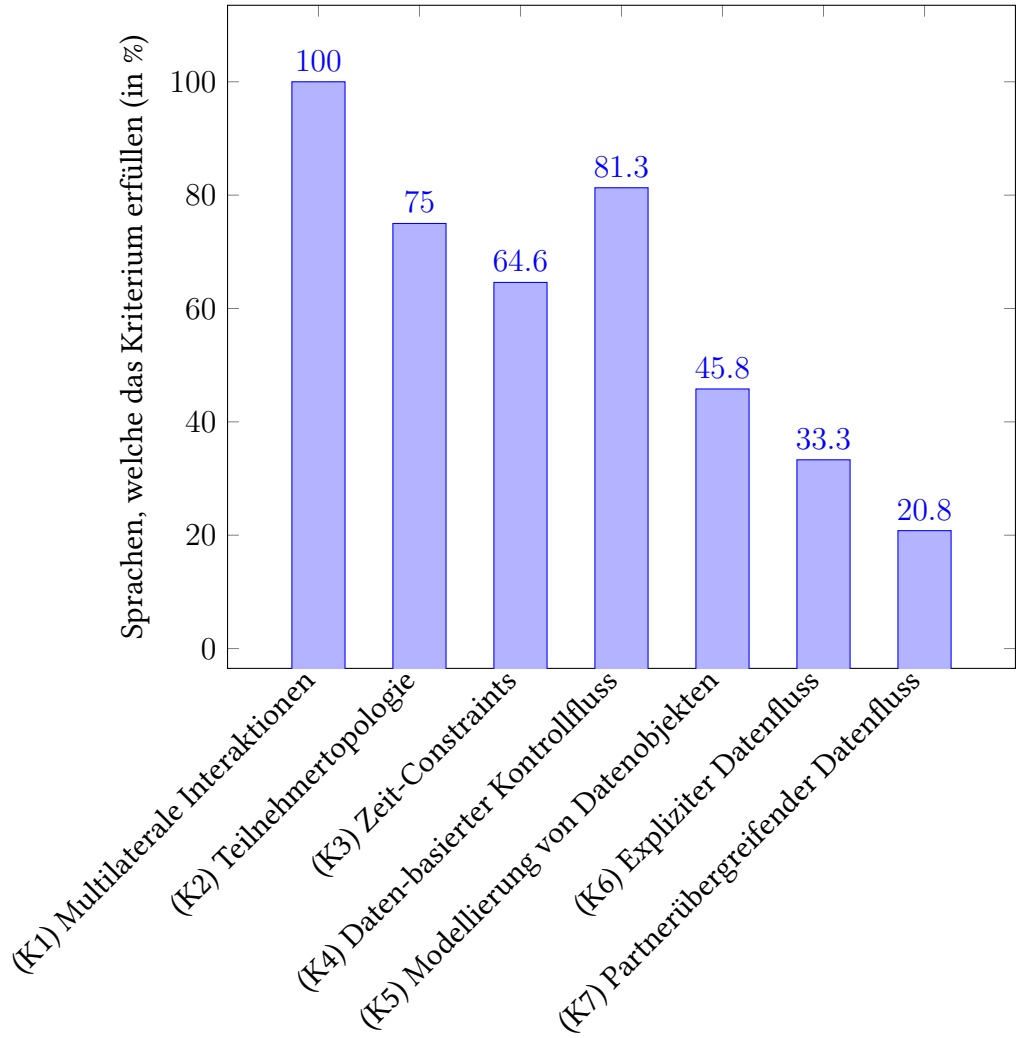



**Abbildung 4.21:** Wieviele Sprachen aus unserer Analyse erfüllen die jeweiligen Kriterien?

# 5 Zusammenfassung und Ausblick

In dieser Arbeit wurde eine Literaturrecherche durchgeführt, bei welcher eine große Menge an möglichen Choreographiesprachen gefunden wurden. Die Basis der Literaturrecherche lieferte hierbei das Ergebnis einer automatisierten, Skript-basierten Suche nach Choreographiesprachen aus [SHKW11] (zum Zeitpunkt der Veröffentlichung dieser Arbeit ist dieses Paper noch nicht veröffentlicht worden). Die resultierende Liste wurde manuell überprüft und anschließend gefiltert, um Sprachen welche nicht dem Choreographie-Konzept entsprechen aus der Liste herauszufiltern. Zusätzlich dazu wurde noch einmal manuell nach Choreographiesprachen gesucht.

Es wurden ebenfalls Kriterien definiert, welche Choreographiesprachen erfüllen sollten. Dabei lag das Hauptaugenmerk auf Kriterien zur Datenmodellierung. Im Anschluss wurden 24 der insgesamt gefundenen 130 Sprachen basierend auf deren Relevanz anhand des definierten Kriterienkatalog analysiert. Die Analyse der jeweiligen Sprachen wurde dabei durch entsprechende Beispiele visuell unterstützt. Bei den Sprachen, die es uns durch frei zugängliche Software ermöglicht haben Beispiele selbst zu modellieren, wurde ein einheitliches Beispiel mittels dieser Sprachen umgesetzt, um einen direkten Vergleich der einzelnen Choreographiesprachen zu ermöglichen.

## Ausblick

Durch die Definition von Kriterien wurde eine Art Richtlinie definiert, welche zukünftig entwickelte Sprachen oder Erweiterungen bereits existierender Sprachen erfüllen sollen. Da im Rahmen einer Bachelorarbeit dieser Kriterienkatalog auf ein paar wichtige Kriterien reduziert werden musste, sollte man diese Kriterien nur als Grundlage sehen. Dieser Katalog sollte daher um weitere Kriterien erweitert werden, um möglichst vielfältige Bereiche und Anwendungsdomänen abzudecken. Da diese Richtlinie für alle Choreographiesprachen gelten soll, es aber einige Choreographiesprachen gibt, welche für spezielle Bereiche angepasst werden, könnte man zusätzlich Kriterienkataloge definieren, welche nur für bestimmte Arten von Choreographien relevant sind. Im Allgemeinen sollte der Choreographie-Begriff deutlicher spezifiziert werden. Durch diese „offene“ Definition, werden relevante Kriterien sehr unterschiedlich definiert, da viele auf bestimmte Punkte mehr wert legen, als auf andere. Das ist auch der Grund, warum Sprachen wie BPEL in einigen Arbeiten zur Modellierung von Choreographien vorgestellt wurden, obwohl diese für Orchestrierungen geeigneter sind und weniger

zur Modellierung von Choreographien. Daher sollte eine eindeutigere Richtlinie spezifiziert werden.

Nachdem diese Richtlinie erweitert wird, kann die Analyse in dieser Arbeit um die neuen Kriterien erweitert werden. Die einzelnen Kriterien könnte man dann ebenfalls gewichten nach der Relevanz dieser Kriterien. Manche Kriterien sollten nämlich erfüllt werden müssen, während andere nur optional und nicht so wichtig sind. Außerdem wurde im Rahmen dieser Arbeit nur eine Teilmenge der gefundenen Choreographiesprachen untersucht. Nur weil diese Sprachen bekannter sind als andere, heißt das aber nicht, dass diese auch besser zur Modellierung von Choreographien geeignet sind. Deshalb sollte die Analyse auch um die noch nicht analysierten Sprachen erweitert werden.

# Literaturverzeichnis


[AAF+02] A. Arkin, S. Askary, S. Fordin, W. Jekeli, K. Kawaguchi, D. Orchard, S. Pogliani, K. Riemer, S. Struble, P. Takacsi-Nagy et al. „Web service choreography interface (WSCI) 1.0“. In: *Standards proposal by BEA Systems, Intalio, SAP, and Sun Microsystems* (2002) (zitiert auf S. 27, 31, 32, 59).

[ACD+03] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte et al. *Business process execution language for web services*. 2003 (zitiert auf S. 28, 51).

[AKM08] F. Arbab, N. Kokash, S. Meng. „Towards Using Reo for Compliance-Aware Business Process Modeling“. In: *Leveraging Applications of Formal Methods, Verification and Validation: Third International Symposium, ISoLA 2008, Porto Sani, Greece, October 13-15, 2008. Proceedings*. Hrsg. von T. Margaria, B. Steffen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, S. 108–123 (zitiert auf S. 26).

[AL05] S. Arroyo, J.-M. López Cobo. „SOPHIE: Architecture and overall algorithm of a choreography service“. In: *Proceeding of First Online Metadata and Semantics Research Conference (MTSR'05)*. 2005, S. 21–30 (zitiert auf S. 26).

[AM15] E. Aarts, M. Mancioppi. „Correction of Unrealizable Service Choreographies“. In: (2015) (zitiert auf S. 19, 23).

[Amb02] S. Ambroszkiewicz. „Entish“. In: *Internet Technologies, Applications and Societal Impact*. Springer, 2002, S. 289–306 (zitiert auf S. 24).

[AP06a] W. M. P. van der Aalst, M. Pesic. „DecSerFlow: Towards a Truly Declarative Service Flow Language“. In: *Web Services and Formal Methods: Third International Workshop, WS-FM 2006 Vienna, Austria, September 8-9, 2006 Proceedings*. Hrsg. von M. Bravetti, M. Núñez, G. Zavattaro. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, S. 1–23 (zitiert auf S. 24, 31, 63).

[AP06b] W. M. van der Aalst, M. Pesic. „Specifying, discovering, and monitoring service flows: Making web services process-aware“. In: *BPM Center Report BPM-06-09, BPMcenter. org* (2006) (zitiert auf S. 63, 64).

[AW01] W. M. P. van der Aalst, M. Weske. „The P2P Approach to Interorganizational Workflows“. In: *Advanced Information Systems Engineering: 13th International Conference, CAiSE 2001 Interlaken, Switzerland, June 4–8, 2001 Proceedings*. Hrsg. von K. R. Dittrich, A. Geppert, M. C. Norrie. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, S. 140–156 (zitiert auf S. 25, 31, 74–76).

[BB09] P. Besana, A. Barker. „An Executable Calculus for Service Choreography“. In: *On the Move to Meaningful Internet Systems: OTM 2009: Confederated International Conferences, CoopIS, DOA, IS, and ODBASE 2009, Vilamoura, Portugal, November 1-6, 2009, Proceedings, Part I*. Hrsg. von R. Meersman, T. Dillon, P. Herrero. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, S. 373–380 (zitiert auf S. 25).

[BBB+02] A. Banerji, C. Bartolini, D. Beringer, V. Chopella, K. Govindarajan, A. Karp, H. Kuno, M. Lemon, G. Pogossiants, S. Sharma et al. „Web services conversation language (wscl) 1.0“. In: *W3C Note* 14 (2002) (zitiert auf S. 27, 32, 59).

[BBC+12] C. Bartolini, A. Bertolino, A. Ciancone, G. De Angelis, R. Mirandola. „Quality Requirements for Service Choreographies.“ In: *WEBIST*. 2012, S. 143–148 (zitiert auf S. 26).

[BCD+05] D. Berardi, D. Calvanese, G. De Giacomo, R. Hull, M. Mecella. „Automatic Composition of Transition-based Semantic Web Services with Messaging“. In: *Proceedings of the 31st International Conference on Very Large Data Bases*. VLDB '05. Trondheim, Norway: VLDB Endowment, 2005, S. 613–624 (zitiert auf S. 23, 31, 70, 71).

[BDDM04] D. Berardi, F. De Rosa, L. De Santis, M. Mecella. „Finite state automata as conceptual model for e-services“. In: *Journal of Integrated Design and Process Science* 8.2 (2004), S. 105–121 (zitiert auf S. 27).

[BDO05] A. Barros, M. Dumas, P. Oaks. „A critical overview of the web services choreography description language“. In: *BPTrends Newsletter* 3 (2005), S. 1–24 (zitiert auf S. 27, 31, 32, 59).

[BDO06] A. Barros, M. Dumas, P. Oaks. „Standards for Web Service Choreography and Orchestration: Status and Perspectives“. In: *Business Process Management Workshops: BPM 2005 International Workshops, BPI, BPD, ENEI, BPRM, WSCOBPM, BPS, Nancy, France, September 5, 2005. Revised Selected Papers*. Hrsg. von C. J. Bussler, A. Haller. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, S. 61–74 (zitiert auf S. 47).

[BGG+05a] N. Busi, R. Gorrieri, C. Guidi, R. Lucchi, G. Zavattaro. „Towards a formal framework for choreography“. In: *14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE'05)*. Juni 2005, S. 107–112 (zitiert auf S. 23).

[BGG+05b] N. Busi, R. Gorrieri, C. Guidi, R. Lucchi, G. Zavattaro. „Choreography and Orchestration: A Synergic Approach for System Design“. In: *Service-Oriented Computing - ICSOC 2005: Third International Conference, Amsterdam, The Netherlands, December 12-15, 2005. Proceedings*. Hrsg. von B. Benatallah, F. Casati, P. Traverso. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, S. 228–240 (zitiert auf S. 22, 32, 86).

[BJR+96] G. Booch, I. Jacobson, J. Rumbaugh et al. „The unified modeling language“. In: *Unix Review* 14.13 (1996), S. 5 (zitiert auf S. 42).

[BKKR03] M. Bernauer, G. Kappel, G. Kramler, W. Retschitzegger. „Specification of Interorganizational Workflows - A Comparison of Approaches“. In: *Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2003)*. International Institute of Informatics und Systemics, 2003, S. 30–36 (zitiert auf S. 18).

[BLWW95] R. W. H. Bons, R. M. Lee, R. W. Wagenaar, C. D. Wrigley. „Modelling interorganizational trade using Documentary Petri Nets“. In: *Proceedings of the Twenty-Eighth Annual Hawaii International Conference on System Sciences*. Bd. 3. Jan. 1995, 189–198 vol.3 (zitiert auf S. 24).

[BRM15] P. Bhuyan, A. Ray, D. P. Mohapatra. „A Service-Oriented Architecture (SOA) Framework Component for Verification of Choreography“. In: *Computational Intelligence in Data Mining - Volume 3: Proceedings of the International Conference on CIDM, 20-21 December 2014*. Hrsg. von L. C. Jain, H. S. Behera, J. K. Mandal, D. P. Mohapatra. New Delhi: Springer India, 2015, S. 25–35 (zitiert auf S. 25).

[BS08] A. K. Bhattacharjee, R. K. Shyamasundar. „ScriptOrc: A Specification Language for Web Service Choreography“. In: *2008 IEEE Asia-Pacific Services Computing Conference*. Dez. 2008, S. 1089–1096 (zitiert auf S. 26).

[BWH09] A. Barker, J. B. Weissman, J. I. van Hemert. „The Circulate architecture: avoiding workflow bottlenecks caused by centralised orchestration“. In: *Cluster Computing* 12.2 (2009), S. 221–235 (zitiert auf S. 23).

[BWR09] A. Barker, C. D. Walton, D. Robertson. „Choreographing Web Services“. In: *IEEE Transactions on Services Computing* 2.2 (Apr. 2009), S. 152–166 (zitiert auf S. 26, 32, 88).

[CCK+01] J. Clark, C. Casanave, K. Kanaskie, B. Harvey, N. Smith, J. Yunker, K. Riemer. „ebXML Business Process Specification Schema Version 1.01“. In: *UN/CEFACT and OASIS* (2001), S. 9 (zitiert auf S. 24).

[CE05a] T. Cottenier, T. Elrad. „Dynamic and decentralized service composition“. In: *Proceedings web information systems and technologies. INSTICC Press* (2005), S. 56–63 (zitiert auf S. 23).

[CE05b] T. Cottenier, T. Elrad. „Executable Choreography Processes with Aspect-Sensitive Services“. In: *Computer Science Department, Illinois Institute of Technology* (2005) (zitiert auf S. 24).

[CFGS10] V. Ciancia, G. L. Ferrari, R. Guanciale, D. Strollo. „Global Coordination Policies for Services“. In: *Electronic Notes in Theoretical Computer Science* 260 (2010), S. 73–89 (zitiert auf S. 26).

[CHY+06] M. Carbone, K. Honda, N. Yoshida, R. Milner, G. Brown, S. Ross-Talbot. „A theoretical basis of communication-centred concurrent programming“. In: *Web Services Choreography Working Group mailing list, to appear as a WS-CDL working report* (2006) (zitiert auf S. 65, 66).

[CHY07] M. Carbone, K. Honda, N. Yoshida. „Structured Communication-Centred Programming for Web Services“. In: *Programming Languages and Systems: 16th European Symposium on Programming, ESOP 2007, Held as Part of the Joint European Conferences on Theory and Practics of Software, ETAPS 2007, Braga, Portugal, March 24 - April 1, 2007. Proceedings*. Hrsg. von R. De Nicola. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, S. 2–17 (zitiert auf S. 25, 31, 65).

[CLNL03] D. W. Cheung, E. Lo, C.-Y. Ng, T. Lee. „Web Services Oriented Data Processing and Integration.“ In: *WWW (Alternate Paper Tracks)*. 2003 (zitiert auf S. 27).

[CS11] A. K. Chopra, M. P. Singh. „Colaba: Collaborative design of cross-organizational processes“. In: *2011 Workshop on Requirements Engineering for Systems, Services and Systems-of-Systems*. Aug. 2011, S. 36–43 (zitiert auf S. 23).

[CT05] I. Chebbi, S. Tata. „CoopFlow: A Framework for Inter-organizational Workflow Cooperation“. In: *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2005, Agia Napa, Cyprus, October 31 - November 4, 2005, Proceedings, Part I*. Hrsg. von R. Meersman, Z. Tari. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, S. 112–129 (zitiert auf S. 23).

[Cyb06] D. Cybok. „A Grid workflow infrastructure“. In: *Concurrency and Computation: Practice and Experience* 18.10 (2006), S. 1243–1254 (zitiert auf S. 25).

[DB08] G. Decker, A. Barros. „Interaction Modeling Using BPMN“. In: *Proceedings of the 2007 International Conference on Business Process Management*. BPM'07. Brisbane, Australia: Springer-Verlag, 2008, S. 208–219 (zitiert auf S. 25, 28).

[DCS09] N. Desai, A. K. Chopra, M. P. Singh. „Amoeba: A Methodology for Modeling and Evolving Cross-organizational Business Processes“. In: *ACM Trans. Softw. Eng. Methodol.* 19.2 (Okt. 2009), 6:1–6:45 (zitiert auf S. 22).

[DGG+15] M. Dalla Preda, M. Gabbrielli, S. Giallorenzo, I. Lanese, J. Mauro. „Dynamic Choreographies“. In: *Coordination Models and Languages: 17th IFIP WG 6.1 International Conference, COORDINATION 2015, Held as Part of the 10th International Federated Conference on Distributed Computing Techniques, DisCoTec 2015, Grenoble, France, June 2-4, 2015, Proceedings*. Hrsg. von T. Holvoet, M. Viroli. Cham: Springer International Publishing, 2015, S. 67–82 (zitiert auf S. 24).

[DKB08] G. Decker, O. Kopp, A. P. Barros. „An introduction to service choreographies (Servicechoreographien – eine Einführung)“. In: *Information Technology* 52.2 (2008), S. 122–127 (zitiert auf S. 13).

[DKLW07] G. Decker, O. Kopp, F. Leymann, M. Weske. „BPEL4Chor: Extending BPEL for Modeling Choreographies“. In: *IEEE International Conference on Web Services (ICWS 2007)*. Juli 2007, S. 296–303 (zitiert auf S. 22, 31, 67).

[DKLW09] G. Decker, O. Kopp, F. Leymann, M. Weske. „Interacting services: From specification to execution“. In: *Data & Knowledge Engineering* 68.10 (2009), S. 946–972 (zitiert auf S. 17, 33).

[DKZD06] G. Decker, M. Kirov, J. M. Zaha, M. Dumas. „Maestro for Let's Dance: An Environment for Modeling Service Interactions“. In: (2006) (zitiert auf S. 80).

[DLC+07] X. Deng, Z. Lin, W. Cheng, R. Xiao, L. Fang, L. Li. „Modeling Web Service Choreography and Orchestration with Colored Petri Nets“. In: *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007)*. Bd. 2. Juli 2007, S. 838–843 (zitiert auf S. 76).

[DMCS04] N. Desai, A. U. Mallya, A. K. Chopra, M. P. Singh. „Protocols+ Policies: A methodology for business process development“. In: *in: Proceedings of the 14th International World Wide Web Conference (WWW'2005)*. Citeseer. 2004 (zitiert auf S. 26).

[DW07] G. Decker, M. Weske. „Local Enforceability in Interaction Petri Nets“. In: *Business Process Management: 5th International Conference, BPM 2007, Brisbane, Australia, September 24-28, 2007. Proceedings*. Hrsg. von G. Alonso, P. Dadam, M. Rosemann. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, S. 305–319 (zitiert auf S. 25).

[DZD06] G. Decker, J. M. Zaha, M. Dumas. „Execution Semantics for Service Choreographies“. In: *Web Services and Formal Methods: Third International Workshop, WS-FM 2006 Vienna, Austria, September 8-9, 2006 Proceedings*. Hrsg. von M. Bravetti, M. Núñez, G. Zavattaro. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, S. 163–177 (zitiert auf S. 80, 81).

[ELT06] J. Eder, M. Lehmann, A. Tahamtan. „Choreographies as Federations of Choreographies and Orchestrations“. In: *Advances in Conceptual Modeling - Theory and Practice: ER 2006 Workshops BP-UML, CoMoGIS, COSS, ECDM, OIS, QoIS, SemWAT, Tucson, AZ, USA, November 6-9, 2006. Proceedings*. Hrsg. von J. F. Roddick, V. R. Benjamins, S. Si-said Cherfi, R. Chiang, C. Claramunt, R. A. Elmasri, F. Grandi, H. Han, M. Hepp, M. D. Lytras, V. B. Mišić, G. Poels, I.-Y. Song, J. Trujillo, C. Vangenot. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, S. 183–192 (zitiert auf S. 24).

[FDVV11] D. Fahland, M. De Leoni, B. F. Van Dongen, W. M. Van Der Aalst. „Many-to-Many: Some Observations on Interactions in Artifact Choreographies.“ In: *ZEUS* 705 (2011), S. 9–15 (zitiert auf S. 13).

[FJ01] U. Frank, J. Jung. „The memo organisation modelling language (memo-orgml)“. In: *Arbeitsberichte des Institut für Wirtschaftsinformatik der Universität Koblenz-Landau* (2001) (zitiert auf S. 25).

[FLB06] J. Fiadeiro, A. Lopes, L. Bocchi. „The SENSORIA Reference Modelling Language: Primitives for Service Description“. In: *available at {www. sensoria-ist. edu}* (2006) (zitiert auf S. 26).

[Fle10] A. Fleischmann. „What Is S-BPM?“ In: *S-BPM ONE – Setting the Stage for Subject-Oriented Business Process Management: First International Workshop, Karlsruhe, Germany, October 22, 2009. Revised Selected Papers*. Hrsg. von H. Buchwald, A. Fleischmann, D. Seese, C. Stary. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, S. 85–106 (zitiert auf S. 26).

[GHB+06] Z. Guan, F. Hernandez, P. Bangalore, J. Gray, A. Skjellum, V. Velusamy, Y. Liu. „Grid-Flow: a Grid-enabled scientific workflow system with a Petri-net-based interface“. In: *Concurrency and Computation: Practice and Experience* 18.10 (2006), S. 1115–1140 (zitiert auf S. 25).

[GHB05] J. Gomez, A. Haller, C. Bussler. „A conversation oriented language for B2B integration based on semantic web services. web service semantics“. In: *Towards dynamic business integration workshop. In conjunction with the 14th international world wide web conference (WWW 2005), Chiba, Japan, May 2005*. 2005 (zitiert auf S. 25).

[HBKL17] M. Hahn, U. Breitenbücher, O. Kopp, F. Leymann. „Modeling and Execution of Data-aware Choreographies. An Overview“. In: *Springer Computer Science - Research and Development (CSRD)* (2017) (zitiert auf S. 33, 34).

[HHL+06] B. Hofreiter, C. Huemer, P. Liegl, R. Schuster, M. Zapletal. „UN/CEFACT'S Modeling Methodology (UMM): A UML Profile for B2B e-Commerce“. In: *Advances in Conceptual Modeling - Theory and Practice: ER 2006 Workshops BP-UML, CoMoGIS, COSS, ECDM, OIS, QoIS, SemWAT, Tucson, AZ, USA, November 6-9, 2006. Proceedings*. Hrsg. von J. F. Roddick, V. R. Benjamins, S. Si-said Cherfi, R. Chiang, C. Claramunt, R. A. Elmasri, F. Grandi, H. Han, M. Hepp, M. D. Lytras, V. B. Mišić, G. Poels, I.-Y. Song, J. Trujillo, C. Vangenot. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, S. 19–31 (zitiert auf S. 27).

[HO06] A. Haller, E. Oren. „m3pl: A Work-FLOWS ontology extension to extract choreography interfaces“. In: *Proceedings of the Workshop on Semantics for Business Process Management (SBPM 2006) at the 3rd European Semantic Web Conference (ESWC 2006). Budva, Montenegro*. Citeseer. 2006 (zitiert auf S. 25).

[HOK06] A. Haller, E. Oren, P. Kotinurmi. „m3po: An Ontology to Relate Choreographies to Workflow Models“. In: *2006 IEEE International Conference on Services Computing (SCC'06)*. Sep. 2006, S. 19–27 (zitiert auf S. 26).

[Int02] P. D. Interface. „XML process definition language“. In: *Document Number WFMC-TC-1025 Document Status-XPDL* 1 (2002) (zitiert auf S. 27).

[JEA+07] D. Jordan, J. Evdemon, A. Alves, A. Arkin, S. Askary, C. Barreto, B. Bloch, F. Curbera, M. Ford, Y. Goland et al. „Web services business process execution language version 2.0“. In: *OASIS standard* 11.120 (2007), S. 5 (zitiert auf S. 67).

[Jen81] K. Jensen. „Coloured petri nets and the invariant-method“. In: *Theoretical Computer Science* 14.3 (1981), S. 317–336 (zitiert auf S. 23, 31, 76).

[JHKK04] J.-y. Jung, W. Hur, S.-H. Kang, H. Kim. „Business process choreography for B2B collaboration“. In: *IEEE Internet Computing* 8.1 (Jan. 2004), S. 37–45 (zitiert auf S. 13).

[JIH10] H. Jamil, A. Islam, S. Hossain. „A declarative language and toolkit for scientific workflow implementation and execution“. In: *International Journal of Business Process Integration and Management* 5.1 (2010), S. 3–17 (zitiert auf S. 22).

[Jun06] J. Jung. *Supply chains in the context of resource modelling*. eng. ICB-Research Report 5. Essen, 2006 (zitiert auf S. 25).

[Kab03] V. Kabilan. „Using multi tier contract ontology to model contract workflow models“. Diss. Data-och systemvetenskap, 2003 (zitiert auf S. 23).

[KEL+11] O. Kopp, L. Engler, T. van Lessen, F. Leymann, J. Nitzsche. „Interaction Choreography Models in BPEL: Choreographies on the Enterprise Service Bus“. In: *Subject-Oriented Business Process Management: Second International Conference, S-BPM ONE 2010, Karlsruhe, Germany, October 14, 2010. Selected Papers*. Hrsg. von A. Fleischmann, W. Schmidt, R. Singer, D. Seese. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, S. 36–53 (zitiert auf S. 22).

[Ker10] N. Kerschbaumer. *Distributed workflow and view definition languages*. Techn. Ber. TR-ISYS, 2010 (zitiert auf S. 24, 27).

[KNMS06] C. Kubczak, R. Nagel, T. Margaria, B. Steffen. „The jABC approach to mediation and choreography“. In: *2nd Semantic Web Service Challenge Workshop*. Bd. 234. Citeseer. 2006 (zitiert auf S. 25).

[Koe03] M. Koethe. „Business Process Definition Metamodel“. In: *Request for Proposals (bei/2003-01-06). Object Management Group* (2003) (zitiert auf S. 23).

[Kop16] O. Kopp. „Partnerübergreifende Geschäftsprozesse und ihre Realisierung in BPEL“. Diss. Stuttgart, Universität Stuttgart, Diss., 2015, 2016 (zitiert auf S. 13).

[KP06] R. Kazhamiakin, M. Pistore. „Choreography Conformance Analysis: Asynchronous Communications and Information Alignment“. In: *Web Services and Formal Methods: Third International Workshop, WS-FM 2006 Vienna, Austria, September 8-9, 2006 Proceedings*. Hrsg. von M. Bravetti, M. Núñez, G. Zavattaro. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, S. 227–241 (zitiert auf S. 47).

[KPR12] D. Knuplesch, R. Pryss, M. Reichert. „Data-aware interaction in distributed and collaborative workflows: Modeling, semantics, correctness“. In: *8th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*. Okt. 2012, S. 223–232 (zitiert auf S. 24).

[KSN92] G. Keller, A.-W. Scheer, M. Nüttgens. *Semantische Prozeßmodellierung auf der GrundlageËreignisgesteuerter Prozeßketten (EPK)"*. Inst. für Wirtschaftsinformatik, 1992 (zitiert auf S. 24, 31, 57).

[KWH07] Z. Kang, H. Wang, P. C. K. Hung. „WS-CDL+: An Extended WS-CDL Execution Engine for Web Service Collaboration“. In: *IEEE International Conference on Web Services (ICWS 2007)*. Juli 2007, S. 928–935 (zitiert auf S. 27).

[KWV+02] S. Krishnan, P. Wagstrom, G. Von Laszewski et al. „GSFL: A workflow framework for grid services“. In: *Preprint ANL/MCS-P980-0802, Argonne National Laboratory* 9700 (2002) (zitiert auf S. 25, 31, 79).

[LAN+13] L. A. F. Leite, G. Ansaldi Oliva, G. M. Nogueira, M. A. Gerosa, F. Kon, D. S. Milojicic. „A systematic literature review of service choreography adaptation“. In: *Service Oriented Computing and Applications* 7.3 (2013), S. 199–216 (zitiert auf S. 18, 19).

[Lee97] R. M. Lee. „A messenger model for navigating among bureaucratic requirements“. In: *Proceedings of the Thirtieth Hawaii International Conference on System Sciences*. Bd. 4. Jan. 1997, 468–477 vol.4 (zitiert auf S. 26).

[Len01] K. Lenz. „Modeling interorganizational workflows with XML nets“. In: *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*. Jan. 2001, 10 pp. (Zitiert auf S. 27).

[Ley+01] F. Leymann et al. *Web services flow language (WSFL 1.0)*. 2001 (zitiert auf S. 27, 31, 51).

[LHPZ06] J. Li, J. He, G. Pu, H. Zhu. „Towards the Semantics for Web Service Choreography Description Language“. In: *Formal Methods and Software Engineering: 8th International Conference on Formal Engineering Methods, ICFEM 2006, Macao, China, November 1-3, 2006. Proceedings*. Hrsg. von Z. Liu, J. He. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, S. 246–263 (zitiert auf S. 23).

[LM08] S. Li, H. Miao. „Modeling the Patterns of WS-CDL Interactions Based on Process Algebra“. In: *2008 International Seminar on Future Information Technology and Management Engineering*. Nov. 2008, S. 222–227 (zitiert auf S. 26).

[LZD12] J. Li, H. Zhang, H. Dun. „A Petri Net Semantics for Web Service in Choreography Description Language“. In: *2012 Sixth International Conference on Internet Computing for Science and Engineering*. Apr. 2012, S. 1–7 (zitiert auf S. 26).

[MBLN09] A. Mahfouz, L. Barroca, R. Laney, B. Nuseibeh. „Requirements-Driven Collaborative Choreography Customization“. In: *Service-Oriented Computing: 7th International Joint Conference, ICSOC-ServiceWave 2009, Stockholm, Sweden, November 24-27, 2009. Proceedings*. Hrsg. von L. Baresi, C.-H. Chi, J. Suzuki. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, S. 144–158 (zitiert auf S. 22).

[MCT09] L. Mei, W. K. Chan, T. H. Tse. „Data Flow Testing of Service Choreography“. In: *Proceedings of the the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering*. ESEC/FSE '09. Amsterdam, The Netherlands: ACM, 2009, S. 151–160 (zitiert auf S. 25).

[MGWH09] M. Milanović, D. Gašević, G. Wagner, M. Hatala. „Rule-Enhanced Business Process Modeling Language for Service Choreographies“. In: *Model Driven Engineering Languages and Systems: 12th International Conference, MODELS 2009, Denver, CO, USA, October 4-9, 2009. Proceedings*. Hrsg. von A. Schürr, B. Selic. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, S. 337–341 (zitiert auf S. 26).

[MH08] J. Mendling, M. Hafner. „From WS-CDL choreography to BPEL process orchestration“. In: *Journal of Enterprise Information Management* 21.5 (2008), S. 525–542 (zitiert auf S. 47).

[Mil06] H. D.-I. N. Milanovic. „Contract-based web service composition“. Diss. Humboldt-Universität zu Berlin, 2006 (zitiert auf S. 23).

[MN04] J. Mendling, M. Nüttgens. „Exchanging EPC business process models with EPML“. In: *XML4BPM* (2004), S. 61–80 (zitiert auf S. 24).

[MNN05] J. Mendling, G. Neumann, M. Nüttgens. „Yet Another Event-Driven Process Chain“. In: *Business Process Management: 3rd International Conference, BPM 2005, Nancy, France, September 5-8, 2005. Proceedings*. Hrsg. von W. M. P. van der Aalst, B. Benatallah, F. Casati, F. Curbera. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, S. 428–433 (zitiert auf S. 27).

[Mod11] B. P. Model. „Notation (BPMN) version 2.0“. In: *OMG Specification, Object Management Group* (2011) (zitiert auf S. 22, 31, 37, 39, 40).

[Mon14] F. Montesi. *Choreographic programming*. InstituttetThe Department, Software u. a., 2014 (zitiert auf S. 23).

[MR97] S. Mauw, M. A. Reniers. „High-level message sequence charts.“ In: *SDL forum*. 1997, S. 291–306 (zitiert auf S. 25).

[MRS05] P. Massuthe, W. Reisig, K. Schmidt. „An Operating Guideline Approach to the SOA“. In: Informatik-Berichte 191. Institut für Informatik, 2005 (zitiert auf S. 26, 31, 81).

[MS03] C. Montangero, L. Semini. „Distributed states temporal logic“. In: *arXiv preprint cs/0304046* (2003) (zitiert auf S. 24).

[MS06] C. Montangero, L. Semini. „A Logical View of Choreography“. In: *Coordination Models and Languages: 8th International Conference, COORDINATION 2006, Bologna, Italy, June 14-16, 2006. Proceedings*. Hrsg. von P. Ciancarini, H. Wiklicky. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, S. 179–193 (zitiert auf S. 24).

[MSLH02] J. Meng, S. Y. W. Su, H. Lam, A. Helal. „Achieving dynamic inter-organizational workflow management by integrating business processes, events and rules“. In: *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*. Jan. 2002, 10 pp. (Zitiert auf S. 24).

[MSW11] A. Meyer, S. Smirnov, M. Weske. *Data in business processes*. 50. Universitätsverlag Potsdam, 2011 (zitiert auf S. 18, 33).

[MY13] F. Montesi, N. Yoshida. „Compositional Choreographies“. In: *CONCUR 2013 – Concurrency Theory: 24th International Conference, CONCUR 2013, Buenos Aires, Argentina, August 27-30, 2013. Proceedings*. Hrsg. von P. R. D'Argenio, H. Melgratti. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, S. 425–439 (zitiert auf S. 23).

[MZW09] T. Motal, M. Zapletal, H. Werthner. „The Business Choreography Language (BCL) - A Domain-Specific Language for Global Choreographies“. In: *2009 World Conference on Services - II*. Sep. 2009, S. 150–159 (zitiert auf S. 22).

[NFH05] B. Norton, S. Foster, A. Hughes. „A Compositional Operational Semantics for OWL-S“. In: *Formal Techniques for Computer Systems and Business Processes: European Performance Engineering Workshop, EPEW 2005 and International Workshop on Web Services and Formal Methods, WS-FM 2005, Versailles, France, September 1-3, 2005. Proceedings*. Hrsg. von M. Bravetti, L. Kloul, G. Zavattaro. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, S. 303–317 (zitiert auf S. 23).

[NLKL07] J. Nitzsche, T. van Lessen, D. Karastoyanova, F. Leymann. „BPELlight“. In: *Business Process Management: 5th International Conference, BPM 2007, Brisbane, Australia, September 24-28, 2007. Proceedings*. Hrsg. von G. Alonso, P. Dadam, M. Rosemann. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, S. 214–229 (zitiert auf S. 22).

[Nor11] A. Norta. „A Choreography Language for eBusiness Collaboration“. In: *Proceedings of the 2011 ACM Symposium on Applied Computing*. SAC '11. TaiChung, Taiwan: ACM, 2011, S. 468–469 (zitiert auf S. 24).

[Oin04] T. Oinn. „XScufl Language Reference“. In: *Internet: Available: www. ebi. ac. uk/tmo/mygrid/XScuflSpecification. html [October 14, 2009]* (2004) (zitiert auf S. 27).

[PE09] G. Pedraza, J. Estublier. „Distributed Orchestration Versus Choreography: The FOCAS Approach“. In: *Trustworthy Software Development Processes: International Conference on Software Process, ICSP 2009 Vancouver, Canada, May 16-17, 2009 Proceedings*. Hrsg. von Q. Wang, V. Garousi, R. Madachy, D. Pfahl. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, S. 75–86 (zitiert auf S. 25).

[PWW+05] S. Parastatidis, J. Webber, S. Woodman, D. Kuo, P. Greenfield. „An introduction to the SOAP service description language“. In: *School of Computing Science, University of Newcastle, Newcastle upon Tyne CS-TR-898* (2005) (zitiert auf S. 26).

[QCP+08] Z. Qiu, C. Cai, L. Peng, X. Zhao, H. Yang. „Reasoning about Channel Passing in Choreography“. In: *2008 2nd IFIP/IEEE International Symposium on Theoretical Aspects of Software Engineering (TASE '08)* 00.undefined (2008), S. 135–142 (zitiert auf S. 23).

[QZCY07] Z. Qiu, X. Zhao, C. Cai, H. Yang. „Towards the Theoretical Foundation of Choreography“. In: *Proceedings of the 16th International Conference on World Wide Web*. WWW '07. Banff, Alberta, Canada: ACM, 2007, S. 973–982 (zitiert auf S. 23, 31, 83).

[RBJ05] J. Rumbaugh, G. Booch, I. Jacobson. „The Unified Modeling Language Reference Manual (with CD-ROM)“. In: (2005) (zitiert auf S. 26, 27, 31, 42, 44, 46, 47, 49).

[RGG96] E. Rudolph, P. Graubmann, J. Grabowski. „Tutorial on Message Sequence Charts“. In: *Computer Networks and ISDN Systems* 28.12 (1996), S. 1629–1641 (zitiert auf S. 26).

[Rob04] D. Robertson. „Multi-agent Coordination as Distributed Logic Programming“. In: *Logic Programming: 20th International Conference, ICLP 2004, Saint-Malo, France, September 6-10, 2004. Proceedings*. Hrsg. von B. Demoen, V. Lifschitz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, S. 416–430 (zitiert auf S. 25).

[RS16] R. Ramanujam, S. Sheerazuddin. „Realizable temporal logics for web service choreography“. In: *Journal of Logical and Algebraic Methods in Programming* 85.5, Part 1 (2016). Special Issue on Automated Verification of Programs and Web Systems, S. 759–781 (zitiert auf S. 26).

[RSF+06] D. Roman, J. Scicluna, D. Fensel, A. Polleres, J. de Bruijn, S. Heymans. *D14v0. 3. Ontology-based Choreography of WSMO Services*. 2006 (zitiert auf S. 27).

[RWR06] S. Rinderle, A. Wombacher, M. Reichert. „Evolution of Process Choreographies in DYCHOR“. In: *On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, GADA, and ODBASE 2006, Montpellier, France, October 29 - November 3, 2006. Proceedings, Part I*. Hrsg. von R. Meersman, Z. Tari. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, S. 273–290 (zitiert auf S. 24).

[SBFZ08] J. Su, T. Bultan, X. Fu, X. Zhao. „Towards a Theory of Web Service Choreographies“. In: *Web Services and Formal Methods: 4th International Workshop, WS-FM 2007, Brisbane, Australia, September 28-29, 2007. Proceedings*. Hrsg. von M. Dumas, R. Heckel. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, S. 1–16 (zitiert auf S. 17, 22, 32, 86).

[Sch11] A. Schönberger. *Do We Need a Refined Choreography Notion?* 2011 (zitiert auf S. 13, 17).

[SHK02] J. Shim, D. Han, H. Kim. „Communication Deadlock Detection of Inter-organizational Workflow Definition“. In: *Databases in Networked Information Systems: Second International Workshop, DNIS 2002 Aizu, Japan, December 16–18, 2002 Proceedings*. Hrsg. von S. Bhalla. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, S. 43–57 (zitiert auf S. 23).

[SHKW11] A. Schönberger, C. Huermer, O. Kopp, A. Wombacher. *List of Choreography Languages – Working Draft*. 2011 (zitiert auf S. 21, 22, 93).

[Sin11] M. P. Singh. „Information-driven Interaction-oriented Programming: BSPL, the Blindingly Simple Protocol Language“. In: *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*. AAMAS '11. Taipei, Taiwan: International Foundation for Autonomous Agents und Multiagent Systems, 2011, S. 491–498 (zitiert auf S. 22).

[STA05] A.-W. Scheer, O. Thomas, O. Adam. „Process modeling using event-driven process chains“. In: *Process-Aware Information Systems* (2005), S. 119–146 (zitiert auf S. 57).

[STDD07] G. van Seghbroeck, F. de Turck, B. Dhoedt, P. Demeester. „Web Service Choreography Conformance Verification in M2M Systems through the piX-model“. In: *IEEE International Conference on Pervasive Services*. Juli 2007, S. 385–390 (zitiert auf S. 26).

[SV05] C. Seel, D. Vanderhaeghen. „Meta-model based extensions of the EPC for inter-organisational process modelling“. In: *Proceedings 4th Workshop on Geschftsprozessmanagement mit Ereignisgesteuerten Prozessketten (EPK 2005)*. Bd. 167. 2005, S. 117–136 (zitiert auf S. 24).

[SW10] A. Schonberger, G. Wirtz. „Towards Executing ebBP-Reg B2Bi Choreographies“. In: *2010 IEEE 12th Conference on Commerce and Enterprise Computing*. Nov. 2010, S. 64–71 (zitiert auf S. 24).

[SWW10] A. Schönberger, C. Wilms, G. Wirtz. *A Requirements Analysis of Business-To-Business Integration*. Techn. Ber. 83. Lehrstuhl für Praktische Informatik, 2010, II, 55 S. : graph. Darst. (Zitiert auf S. 17).

[SXS12] Y. Sun, W. Xu, J. Su. „Declarative Choreographies for Artifacts“. In: *Service-Oriented Computing: 10th International Conference, ICSOC 2012, Shanghai, China, November 12-15, 2012. Proceedings*. Hrsg. von C. Liu, H. Ludwig, F. Toumani, Q. Yu. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, S. 420–434 (zitiert auf S. 24).

[SY07] H. Sun, J. Yang. „BTx-Net: A Token Based Dynamic Model for Supporting Consistent Collaborative Business Transactions“. In: *IEEE International Conference on Services Computing (SCC 2007)*. Juli 2007, S. 490–497 (zitiert auf S. 23).

[SY08] H. Sun, J. Yang. „CoBTx-Net: A Model for Reliability Verification of Collaborative Business Transaction“. In: *Business Process Management Workshops: BPM 2007 International Workshops, BPI, BPD, CBP, ProHealth, RefMod, semantics4ws, Brisbane, Australia, September 24, 2007, Revised Selected Papers*. Hrsg. von A. ter Hofstede, B. Benatallah, H.-Y. Paik. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, S. 220–231 (zitiert auf S. 23).

[SZS06] Y. Shi, L. Zhang, B. Shi. „TDWF: A Workflow Model Based on Cooperation of Web Services“. In: *2006 10th International Conference on Computer Supported Cooperative Work in Design*. Mai 2006, S. 1–6 (zitiert auf S. 26).

[SZZ+12] W. Song, G. Zhang, Y. Zou, Q. Yang, X. Ma. „Towards Dynamic Evolution of Service Choreographies“. In: *2012 IEEE Asia-Pacific Services Computing Conference*. Dez. 2012, S. 225–232 (zitiert auf S. 26).

[Tha01] S. Thatte. „XLANG: Web services for business process design“. In: *Microsoft Corporation* 2001 (2001) (zitiert auf S. 28).

[Van98] W. M. Van der Aalst. „The application of Petri nets to workflow management". In: *Journal of circuits, systems, and computers* 8.01 (1998), S. 21–66 (zitiert auf S. 74, 81).

[VIG+10] H. Vincent, V. Issarny, N. Georgantas, E. Francesquini, A. Goldman, F. Kon. „CHOReOS: Scaling Choreographies for the Internet of the Future". In: *Middleware '10 Posters and Demos Track*. Middleware Posters '10. Bangalore, India: ACM, 2010, 8:1–8:3 (zitiert auf S. 23).

[VMP+12] V. Valero, H. Macià, J. J. Pardo, M. E. Cambronero, G. Díaz. „Transforming Web Services Choreographies with priorities and time constraints into prioritized-time colored Petri nets". In: *Science of Computer Programming* 77.3 (2012). Feature-Oriented Software Development (FOSD 2009), S. 290–313 (zitiert auf S. 26).

[VSC04] P. Villarreal, E. Salomone, O. Chiotti. „A UML Profile for Modeling Collaborative Business Processes based on Interaction Protocols". In: *Argentine Symposium on Information Systems (ASIS 2004)*. Bd. 33. 2004 (zitiert auf S. 27).

[WFMN04] A. Wombacher, P. Fankhauser, B. Mahleko, E. Neuhold. „Matchmaking for business processes based on choreographies". In: *e-Technology, e-Commerce and e-Service, 2004. EEE '04. 2004 IEEE International Conference on*. März 2004, S. 359–368 (zitiert auf S. 24, 32, 84).

[Whi04] S. A. White. „Introduction to BPMN". In: *IBM Cooperation* 2 (2004) (zitiert auf S. 22, 31, 37).

[WRS+09] S. Wieczorek, A. Roth, A. Stefanescu, V. Kozyura, A. Charfi, F. M. Kraft, I. Schieferdecker. „Viewpoints for modeling choreographies in service-oriented architectures". In: *2009 Joint Working IEEE/IFIP Conference on Software Architecture European Conference on Software Architecture*. Sep. 2009, S. 11–20 (zitiert auf S. 25).

[XOWY10] Y. Xia, Z. Ouyang, Y. Wu, R. Yang. „Determing performance of choreography-based composite services". In: *Journal of Convergence Information Technology* 5.8 (2010) (zitiert auf S. 25).

[XQH+07] D. H. Xu, Y. Qi, D. Hou, Y. Chen, L. Liu. „A Formal Model for dynamic Web Services Composition MAS-Based and Simple Security Analysis Using Spi Calculus". In: *Next Generation Web Services Practices, 2007. NWeSP 2007. Third International Conference on*. Okt. 2007, S. 69–72 (zitiert auf S. 23).

[Yeu06] W. l. Yeung. „Mapping WS-CDL and BPEL into CSP for Behavioural Specification and Verification of Web Services". In: *2006 European Conference on Web Services (ECOWS'06)*. Dez. 2006, S. 297–305 (zitiert auf S. 23).

[YLZ11] S. Yongchareon, C. Liu, X. Zhao. „An Artifact-Centric View-Based Approach to Modeling Inter-organizational Business Processes". In: *Web Information System Engineering – WISE 2011: 12th International Conference, Sydney, Australia, October 13-14, 2011. Proceedings*. Hrsg. von A. Bouguettaya, M. Hauswirth, L. Liu. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, S. 273–281 (zitiert auf S. 22).

[YRB+10] Q. Yu, M. Rege, A. Bouguettaya, B. Medjahed, M. Ouzzani. „A two-phase framework for quality-aware Web service selection“. In: *Service Oriented Computing and Applications* 4.2 (2010), S. 63–79 (zitiert auf S. 27).

[YW03] S.-B. Yan, F.-J. Wang. „A Cooperative Framework for Inter-Organizational Workflow System“. In: *2013 IEEE 37th Annual Computer Software and Applications Conference* 00.undefined (2003), S. 64 (zitiert auf S. 23).

[YZQ+06] H. Yang, X. Zhao, Z. Qiu, G. Pu, S. Wang. „A Formal Model for Web Service Choreography Description Language (WS-CDL)“. In: *2006 IEEE International Conference on Web Services (ICWS'06)*. Sep. 2006, S. 893–894 (zitiert auf S. 23).

[ZBDH06] J. M. Zaha, A. Barros, M. Dumas, A. ter Hofstede. „Let's Dance: A Language for Service Behavior Modeling“. In: *On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, GADA, and ODBASE 2006, Montpellier, France, October 29 - November 3, 2006. Proceedings, Part I*. Hrsg. von R. Meersman, Z. Tari. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, S. 145–162 (zitiert auf S. 25, 31, 80).

[ZZ09] J. Zhou, G. Zeng. „A mechanism for grid service composition behavior specification and verification“. In: *Future Generation Computer Systems* 25.3 (2009), S. 378–383 (zitiert auf S. 23, 25).

Alle URLs wurden zuletzt am 01.06.2017 geprüft.

**Erklärung**

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

---

Ort, Datum, Unterschrift