Hochschule Neu-Ulm
University of Applied Sciences

Bachelorarbeit
im Bachelorstudiengang
Information Management Automotive
an der Hochschule für angewandte Wissenschaften Neu-Ulm

Konzeptionierung und Modellierung einer BPMN-Erweiterung für flexible Geschäftsprozesse

Erstkorrektor/-in: Prof. Dr. Thomas Bauer
Betreuer/-in: *Prof. Dr. Jörg-Oliver Vogt*

Verfasser/-in: Hai Duy Pham (Matrikel-Nr.: 293539)

Thema erhalten: 27.06.2025
Arbeit abgegeben: 27.10.2025

Inhalt

Abkürzungsverzeichnis

Abb.	Abbildung
BPMN	Business Process Model and Notation
BPMN+X	BPMN plus Extension
CDME	Conceptual Domain Model of the Extension
CoPMoF	Controlable Pre-Modeled Flexibility
MDA	Model-Driven Architecture
GLSP	Eclispe Graphical Language Server Plattform
GP	Geschäftsprozess
OMG	Object Management Group
UML	Unified Modeling Language
WSL	Windows Subsystem for Linux
XML	Extensible Markup Language
XSD	Extension Definition Document (XSD)

1. Einleitung

Die Modellierung von Geschäftsprozessen ist ein zentrales Instrument zur Analyse, Optimierung und Automatisierung von betrieblichen Abläufen. Bei Business Process Model and Notation (BPMN) 2.0 handelt es sich um einen international etablierten Standard. Der Standard der Object Management Group (OMG) hat sich im Laufe der Zeit durchgesetzt. BPMN bietet eine einheitliche Notation zur Darstellung von Prozessabläufen und wird in Forschung und Praxis gleichermaßen eingesetzt. Sie dient als Brücke zwischen fachlichen Beschreibungen und technischen Ausführungen. Trotz des hohen Verbreitungsgrades des BPMN-Standards stößt diese Notation bei der Modellierung domänenspezifischen oder komplexeren Sachverhalten jedoch an ihre Grenzen. Insbesondere fehlt es an Mechanismen, um bestimmte Aspekte – etwa Flexibilität, Perspektiven oder domänenspezifische Konzepte – strukturiert und standardkonform darzustellen.

Zur Schließung dieser Lücke wurden in den letzten Jahren verschiedene methodische Ansätze entwickelt, um die Erweiterung von BPMN auf formaler Ebene ermöglichen. Einen grundlegenden Beitrag hierzu leisteten Stroppi et al. (2011) mit der Entwicklung einer modellgetriebenen Vorgehensweise, dem Model-Driven Architecture (MDA) Ansatz. Diese soll den Prozess von der konzeptionellen Modellierung bis zu technischen Implementierungen in vier klar definierte Phasen strukturieren. Darauf aufbauend erweiterten Braun und Esswein (2015) die Methode um die Möglichkeit, mehrperspektivische Sichten (Perspectives) und zusätzliche Diagrammtypen in BPMN zu integrieren. Ihr Ansatz zielt darauf ab, die zunehmende Komplexität großer Prozessmodelle durch perspektivische Trennung und die Erweiterung des BPMN-Metamodells um Perspective- und Diagram-Klassen beherrschbar zu machen diese methodische Grundlage wurde in den letzten Jahren von Bauer (2017–2025) im Rahmen der Forschungsreihe CoPMoF – Controllable Pre-Modeled Flexibility weitergeführt und konkretisiert.

Vor dem Hintergrund dieser theoretischen und methodischen Grundlagen verfolgt die vorliegende Arbeit das Ziel, die Konzepte der BPMN-Erweiterbarkeit im Kontext praktischer Entwicklungsumgebungen zu untersuchen und exemplarisch umzusetzen. Im Mittelpunkt steht dabei die Entwicklung und Evaluation einer technischen Erweiterung auf Basis des Open-BPMN-Frameworks, die darauf abzielt, domänenspezifische Modellierungskonzepte in BPMN 2.0 zu integrieren. Hierzu wird zunächst der methodische Ansatz zur Erweiterung des bestehenden Metamodells beschrieben. Anschließend wird die technische Implementierung innerhalb der Open-BPMN-Architektur vorgestellt. Ein besonderer Fokus liegt auf der

Einrichtung und Konfiguration der Entwicklungsumgebung sowie den dabei auftretenden Herausforderungen, insbesondere im Zusammenhang mit den verwendeten Werkzeugen und Betriebssystemumgebungen.

2. Theoretischer Hintergrund

2.1 Business Process Model and Notation

2.1.1 BPMN 2.0

BPMN ist ein Modelierungsstandard der von der OMG entwickelt worden ist. Das Hauptziel besteht darin, Notationen zu bieten, die sich durch eine intuitive Verständlichkeit auszeichnen und von allen betroffenen Personen verstanden werden können. Die BPMN stellt eine Verbindung zwischen der Geschäftsprozessmodellierung und der Geschäftsprozessimplementierung her. Ein weiteres Ziel von BPMN ist die Visualisierung von Extensible Markup Languages (XML), die zur Ausführung von Geschäftsprozessen entworfen worden sind, durch geschäftsorientierte Notationen. (OMG, 2011, S. 1)

Die BPMN repräsentiert eine Zusammenführung bewährter Verfahren der Geschäftsmodellierung. Die Methoden fanden Anwendung bei der Definition der Notation und Semantik von Kollaborations-, Prozess- und Choreografie-diagrammen. Die Die Intention von BPMN besteht darin, die Standardisierung der Geschäftsprozessmodellierung und -notation angesichts einer Vielzahl diverser Modellierungsnotationen und Sichtweisen zu gewährleisten. (OMG, 2011, S. 1)

Das BPMN definiert die verschiedenen Arten von Konformität als Prozessmodellierungskonformität, Prozessausführungskonformität, Buisness Process Execution Language-Prozessausführungskonformität und Choreografiemodellierungs-konformität. Es sei darauf hingewiesen, dass eine Implementierung lediglich einer dieser Konformitäten aufweisen muss, um den BPMN-Standard zu erfüllen. Im Rahmen der vorliegenden Arbeit erfolgt eine detaillierte Betrachtung der Prozessmodellierungskonformität. (OMG, 2011, S. 2)

2.1.2 BPMN-Prozessmodellierungskonform

Implementierungen, die den Anspruch erheben, BPMN-konform zu sein, müssen die folgenden BPMN-Pakete unterstützen, nämlich die BPMN-Kernpakete mit den in Infrastructure, Foundation, Common und Service definierten Elementen, Prozessdiagramme mit den in Process, Activities, Data und Human Interaction beschriebenen Elementen, Kollaborationsdiagramme einschließlich Pools und Message Flow sowie Konversationsdiagramme einschließlich Pools, Conversations und Conversation Links. (OMG, 2011, S. 2)

Im Rahmen der angestrebten Prozessmodellierungskonformität wurden alternativ die drei Konformitätsunterklassen Descriptive, Analytic und Common Executable definiert. Es besteht die Möglichkeit, eine Erweiterung zu implementieren, die lediglich eine der drei Unterklassen unterstützt. Die Unterklasse Descriptive befasst sich mit sich mit visuellen Elementen und Attributen. Die Klasse Analytic umfasst den Inhalt der Klasse Descriptive und beinhaltet die Hälfte der Konstrukte einer vollständigen Prozessmodellierungskonformität. Die beiden Unterklassen fokussieren sich auf einsehbare Elemente und vernachlässigen unterstützende Attribute und Elemente fast vollständig. Der Fokus von Common Executable liegt auf den Elementen, die für die Modellierung ausführbarer Prozesse erforderlich sind. Damit ein Programm eine Unterklasse unterstützt, muss es zwingend alle Elemente der Unterklasse sowie deren gelisteten Attribute unterstützen. Elemente und Attribute welche sich nicht in den drei Unterklassen befinden sind in der vollständigen Prozessmodellierungskonformität-Klasse. (OMG, 2011, S. 2-3)

Ein wesentlicher Aspekt der BPMN sind die Formen und Symbole, die für ihre grafische Darstellung genutzt werden. Die Generierung einer Visualisierungssprache, die eine breite Verständlichkeit aufweist, ist von signifikanter Relevanz. Daher ist es essenziell, dass Implementierungen, welche zur Erstellung von BPMN-Prozessdiagrammen genutzt werden, die von BPMN vordefinierten grafischen Elemente verwenden. Es sei an dieser Stelle angemerkt, dass Größe, Farbe, Linienart sowie Schriftposition flexibel sind, es sei denn, es wird explizit darauf hingewiesen. Des Weiteren besteht die Möglichkeit, das BPMN-Diagramm durch neue Elemente zu erweitern. Die Hinzufügung von neuen Markierungen und Indikatoren ist möglich. Die spezifischen Attribute von BPMN-Elementen können durch deren Ergänzung betont werden. Die Modifikation von Linienarten ist unter der Voraussetzung zulässig, dass keine Kollision mit vordefinierten Linienarten eintritt. Die Modifikation der Farbgebung von Objekten ist zulässig. Dies kann die Implikation spezifischer Semantiken oder die Übertragung zusätzlicher Informationen bewirken. Darüber hinaus ist es untersagt, Erweiterungen in einer Weise zu gestalten, dass die Form eines bereits definierten grafischen Elements beeinflusst wird.

Eine Implementierung, die BPMN-Diagramme generiert und darstellt, hat die Spezifikationen und Beschränkungen für Verbindungen sowie andere schematische Beziehungen zwischen grafischen Elementen zwingend einzuhalten. Sind zulässige oder geforderte Verbindungen als bedingt und von Attributen der jeweiligen Konzepte abhängig definiert, so ist die

Implementierung dazu verpflichtet, die Konsistenz zwischen den hergestellten Verbindungen und den jeweiligen Attributwerten sicherzustellen.

Für eine konforme Implementation muss die Semantik von Diagrammelementen unterstützt werden. Hierfür werden im BPMN viele semantische Konzepte von Processes definiert und an grafischen Elementen assoziiert.

Eine konforme Implementierung ist nicht verpflichtet, Elemente oder Attribute zu unterstützen, die in der Spezifikation als nicht normativ oder informativ klassifiziert sind. Derartige Elemente werden als optional gekennzeichnet. Zudem wird definiert, auf welche Art und Weise, in welcher Form und in welchem Ausmaß das Element angezeigt wird und ob es unterstützt wird. (OMG, 2011, S. 9)

Eine weitere Anforderung, die es zu erfüllen gilt, ist die Bereitstellung eines Austauschformats, um die Implementierung in verschiedenen Tools zu ermöglichen. Darüber hinaus wird empfohlen, die Implementation mit dem Process-Metamodell zu koordinieren. Die vorliegende Integration ermöglicht die Implementierung der Prozessdiagramme in diversen Herstellerumgebungen. (OMG, 2011, S. 9)

2.1.3 BPMN-Erweiterungsmechanismus

Die BPMN 2.0 beinhaltet einen Erweiterungsmechanismus, der es gestattet, BPMN-Elemente mit neuen Attributen zu versehen. Darüber hinaus besteht die Möglichkeit, neue Elemente zu implementieren, ohne dabei den BPMN-Standard zu verletzen. Die Attribute der Erweiterung dürfen der Semantik von jeglichen BPMN-Elementen nicht widersprechen. Des Weiteren ist es essenziell, dass das daraus resultierende Diagramm von den Lesern intuitiv verstanden werden kann. Aus diesem Grund ist es essenziell, dass die Grundstruktur grundlegender Flusselemente einer BPMN-Konformen Erweiterung nicht modifiziert wird. Dies bezieht sich auf Gateways, Ereignissen und Aktivitäten. (OMG, 2011, Seite 44)

Das Metamodell beinhaltet eine Reihe von Erweiterungselementen, die es den Anwendern ermöglichen, neue Attribute und Elemente zu den bereits existierenden BPMN-Elementen hinzuzufügen. Eine BPMN-Erweiterung setzt sich demnach aus den vier Elementen Extension, ExtensionDefinition, ExtensionAttributeDefinition und ExtensionAttributeValue zusammen. Es sei darauf hingewiesen, dass die Elemente ExtensionAttributes und ExtensionDefinition von besonderer Relevanz sind. Gemäß der Definition des ExtensionAttributeDefinition wird eine Liste von Attributen beschrieben, die an jedes BPMN-Element angefügt werden kann. Die vorliegende Liste spezifiziert den Namen sowie den Typ der neuen Attribute. Dadurch

wird es Anwendern ermöglicht, beliebige Metamodelle in ein BPMN-Metamodell zu integrieren und die vorhandenen Elemente wiederzuverwenden. Die Erstellung der ExtensionDefinition kann dabei vollständig unabhängig von BPMN-Elementen und -Definitionen erfolgen. Das Element Extension bindet ein ExtensionDefinition und seine Werte an eine BPMN-Model definition. Die ExtensionAttributeDefinition dient der Definition neuer Attribute, deren Werte in ExtensionAttributeValue gespeichert werden. (OMG, 2011, Seite 57-69)

2.2 Unified Modeling Language

Die Unified Modeling Language (UML) ist eine Modelierungssprache, die zur Spezifizierung, Visualisierung, Konstruktion und Dokumentation der Artefakte von Softwaresystemen sowie zur Geschäftsmodellierung und für andere Nicht-Softwaresysteme eingesetzt werden kann. Die UML repräsentiert eine Sammlung bewährter Verfahren aus dem Ingenieurwesen, die sich bei der Modellierung großer und komplexer Systeme als effektiv erwiesen haben. (OMG, 2017, Seite 44)

UML wurde aus den drei führenden objektorientierten Methoden Object Oriented Software Engineering, Object-Modeling Technique und dem Object-Oriented Analysis and Design kreiert. Darüber hinaus wurden eine Reihe bewährter Verfahren aus den Disziplinen Modellierungssprachen-Design, objektorientierte Programmierung und Architekturbeschreibungssprachen integriert. Im Zuge ihrer Entwicklung erfuhr UML eine signifikante Erweiterung, die sich in präziseren Definitionen seiner abstrakten Syntaxregeln und Semantik, einer modulareren Sprachstruktur sowie einer erheblich verbesserten Fähigkeit zur Modellierung groß angelegter Systeme zeigte. (OMG, 2017, Seite 44)

Ein primäres Ziel von UML besteht darin, den Stand der Technik in der Branche durch die Interoperabilität von visuellen Modellierungswerkzeugen für Objekte zu fördern. Für die Realisierung dieses Vorhabens ist die Verwendung einer einheitlichen Semantik und Syntax erforderlich. Für eine UML-Notation hat OMG folgende Anforderungen aufgestellt. (OMG, 2017, Seite 44)

Die Spezifikation der abstrakten Syntax eines UML erfordert die Definition eines gemeinsamen, auf Meta Object Facility basierenden Metamodells. Die abstrakte Syntax dient der Definition von UML-Modellierungskonzepten, ihren Attributen sowie deren Beziehungen untereinander. Darüber hinaus werden durch sie Regeln für die Kombination von Konzepten definiert, die zur Erstellung partieller oder vollständiger Um-Modelle herangezogen wird.

Die abstrakte Syntax definiert die Menge der UML-Modellierungskonzepte, ihre Attribute und ihre Beziehungen sowie die Regeln für die Kombination dieser Konzepte zur Erstellung partieller oder vollständiger UML-Modelle. (OMG, 2017, Seite 44)

UML-Modellierungskonzepte benötigen eine detaillierte Erläuterung ihrer Semantik. Diese muss in einer technologieunabhängigen Weise definiert, wie das UML-Konzept von Computern umgesetzt werden sollen. (OMG, 2017, Seite 44)

Eine detaillierte Erläuterung der Semantik jedes UML-Modellierungskonzepts ist erforderlich. Die Semantik definiert auf technologieunabhängige Weise die Umsetzung von UML-Konzepten durch Computer. (OMG, 2017, Seite 44)

Es wird eine Spezifikation der menschenlesbaren Notationselemente benötigt. Diese legt fest, wie die einzelnen UML-Modellierungskonzepte visuell darzustellen sind und nach welchen Regeln sie zu unterschiedlichen Diagrammtypen kombiniert werden. Die Auswahl der Diagrammtypen erfolgt unter Berücksichtigung diverser Aspekte der modellierten Systeme.

(OMG, 2017, Seite 44)

2.3 Stroppi-Methode

Im Jahr 2011 erfolgte die Veröffentlichung der Stroppi Methode durch Luis Jesús Ramon Stroppi, Omar Chiotti und Pablo David Villarreal. Die Methode beschäftigt sich mit einer MDA, die zur Entwicklung von Erweiterungen für das Metamodell von BPMN 2.0 vorgeschlagen wird. Im Rahmen des vorliegenden Ansatzes erfolgt eine Erläuterung der auf MDA basierenden Methode. (Stroppi et. al, 2011, S.59-60)

Der Ansatz umfasst die Konzeption der Erweiterung unter Verwendung von UML, ihre grafische Darstellung unter Berücksichtigung des BPMN-Erweiterungsmechanismus und ihre Transformation in XML-Schema-Dokumente, die von BPMN-konformen Werkzeugen verarbeitet werden können. (Stroppi et. al, 2011, S.59-60)

Zu diesem Zweck wird zunächst ein Conceptual Domain Model of the Extension (CDME) durch UML definiert. Im Anschluss erfolgt die Erstellung eines BPMN plus Extension (BPMN+X) Modells, welches die Erweiterung anhand des BPMN-Erweiterungsmechanismus beschreibt. Im darauffolgenden Schritt erfolgt die Transformation des BPMN+X-Modells in ein XML-Schema Extension Definition Model. In der finalen Phase des Prozesses erfolgt die Transformation des XML Schema Extension Definition Models in ein XML Schema

Extension Definition Document (XSD). Im folgenden Abschnitt wird mehr auf die einzelnen Schritte des Ansatzes eingegangen. (Stroppi et. al, 2011, S.63)

Die initiale Handlung im Rahmen der Methode besteht in der Erstellung eines CDME in Form eines UML-Klassendiagramms. Das UML-Klassendiagramm veranschaulicht die neu definierten Elemente und Attribute der Erweiterung. Darüber hinaus erfolgt eine Verknüpfung der neuen Elemente mittels Assoziationen zu existierenden BPMN-Klassen. Die Nutzung von UML ermöglicht die Umgehung von Einschränkungen des BPMN-Erweiterungsmechanismus. (Stroppi et. al, 2011, S.63-64)

Im zweiten Schritt der Methode wird ein BPMN+X-Model als UML-Profil aus dem zuvor generierten CDME definiert. BPMN+X ist eine für die Methode entwickelte Modellierungssprache. Ihre Semantik und abstrakte Syntax basiert auf einer Spezifikation des BPMN-Erweiterungsmechanismus. Zunächst wird der Container ExtensionModel erstellt. Dieser ist höhergestellt als alle Elemente, die die BPMN-Erweiterung definieren. Elemente des BPMN-Metamodells werden als BPMNElement dargestellt. Die Container BPMNEnum und ExtensionEnum repräsentieren Mengen von Werten, die im BPMN-Metamodell und Erweiterungsmodell definiert worden sind. Neue Elemente der Erweiterung werden als ExtensionElement repräsentiert. Eine Gruppe von Attributen, die namentlich versehen ist und BPMN-Elemente erweitert, wird als ExtensionDefinition dargestellt. Eine konzeptionelle Verbindung zwischen einem BPMNElement und einer ExtensionDefinition wird durch ein ExtensionRelationship dargestellt. (Stroppi et. al, 2011, S.64-65)

Im dritten Schritt erfolgt die Transformation eines BPMN+X-Modells in ein XML-Schema-Extension-Definition-Modell. Zu diesem Zweck werden ExtensionModel Elemente in Schema Elemente, ExtensionDefinition Elemente in ModelGroupDefinition Elemente, ExtensionElement Elemente in ModelGroupDefinition Elemente und ExtensionEnum Elemente in SimpleTypeDefinition Elemente umgewandelt. Es ist nicht notwendig, BPMNElement- und BPMNEnum-Elemente umzuwandeln, da das generierte Schema BPMN-Spezifikationen importiert. Die Referenzierung von BPMN-Elementen durch Elemente des ExtensionModels ist demnach möglich. (Stroppi et. al, 2011, S.69-70)

Im finalen Schritt erfolgt die Transformation des XML-Schema-Extension-Definition-Modells in ein XSD. Für diesen Prozess wird eine sogenannte Model-to-Code-Transformation durchgeführt. Im vorliegenden Prozess wird jedes Element in ein Element umgewandelt, welches den Vorgaben des XSD entspricht. Schema-Elemente werden in ein <xsd:schema>-

Element, GroupDefinition-Elemente werden in ein <xsd:group>-Element, ComplexTypeDefinition-Elemente werden in ein <xsd:complexTypeDefinition>-Element und SimpleTypeDefinition-Elemente werden in <xsd:simpleTypeDefinition>-Elemente umgewandelt. Das daraus resultierende Dokument wird von den Autoren XML Schema Extension Definition Document genannt. (Stroppi et. al, 2011, S. 70) Es sei darauf hingewiesen, dass die Elemente <xsd:complexTypeDefinition> und <xsd:simpleTypeDefinition> keine gültigen Elemente einer XSD sind. Daher wird angenommen, dass mit den verwendeten Begriffen <xs:complexType> und <xs:complexType> gemeint sind.

2.4 Controlable Pre-Modeled Flexibility

Ziel von CoPMoF ist es Geschäftsprozesse flexibler zu gestalten. Hierfür wird die nötige Flexibilität bereits beim Erstellen des Modells durch vormodellierte Elemente erreicht. Zur Erreichung des zuvor definierten Ziels wurden unterschiedliche Ansatzpunkte hinsichtlich der Aspekte Kontrollflüsse und Geschäftsprozesse skizziert. Dies geschieht durch die Verwendung von optionalen Aktivitäten und Kanten, Dynamischen Sprüngen, alternativen Kantenarten und kritischer Auswahlverfahren. Dadurch kann eine optionale, aber gewünschte Ausführungsreihenfolge von Aktivitäten vorbestimmt werden. Durch optionale Aktivitäten und Kanten kann man frühzeitig Prozesselemente starten oder auch erst nach einer Mindestlaufzeit der voranliegenden Aktivität. Hierbei ist zu anzumerken das, die optionalen Aktivitäten nur in Hinsicht ihrer Startbedingungen optionale Eigenschaften besitzen und weiterhin obligatorisch bleiben für den GP. Die parallele Modellierung von Aktivitäten ist mit CoPMoF um einiges übersichtlicher und kann intuitiv von Endbenutzer erkannt werden im Gegensatz zu herkömmlichen PMS. (Bauer, 2022, S. 1; Bauer, 2023a, S. 1) Gemäß dem aktuellen Stand der Forschung ist eine parallele Modellierung von Aktivitäten, die mit optionalen Kanten verbunden sind, in einem handelsüblichen PMS unter Berücksichtigung der gewünschten Ausführungsreihenfolge möglich. Diese Vorgehensweise ist jedoch als unübersichtlich zu erachten und für den Endbenutzer nicht intuitiv erkennbar. (Bauer, 2025, S. 804-805)

2.4.1 Anforderungen

Im Folgenden werden die von Bauer definierten Anforderungen für vormodellierte Flexibilität dargelegt. Die in diesem Kontext formulierten Anforderungen werden von der BPMN nicht ausreichend abgedeckt. Alternativ ist die Realisierung dieser Anforderungen nur durch komplexe Prozessgraphen möglich. Der Fokus von Bauers Ansatz liegt auf der Bereitstellung

einer intuitiven Modellierungsumgebung für GP-Designer. Die vorliegende Umgebung erlaubt die Gestaltung von Flexibilität bereits in der Planungsphase. Die Realisierung erfolgt durch die Integration von Konstrukten, die durch das Modellierungswerkzeug bereitgestellt werden. In diesem Zusammenhang präsentiert Bauer optionale Aktivitäten und Kanten, spezielle Kantentypen, dynamische Sprünge sowie gegenseitige Ausschließungen. Die hier beschriebene Form der Flexibilität steht den Nutzern während der Laufzeit eines GP zur Verfügung. Diese Entwicklung lässt sich auf die bereits vormodellierten Eigenschaften der Elemente zurückführen.

2.4.1.1 Optionale Aktivitäten und optionale Kanten

Aktivitäten können als optional gekennzeichnet werden. Diese können übersprungen werden, sofern sie keine Relevanz für den GP aufweisen. Im Rahmen der Laufzeit des GP werden potenziellen Arbeitskräften optionale Aktivitäten in ihrer Aufgabenliste angezeigt. Die Entscheidung über den Abschluss der Aktivität oder deren Überspringen liegt in diesem Fall beim Nutzer. In Situationen, in denen ein Zeitmangel herrscht, empfiehlt es sich, von optionalen Aktivitäten abzusehen. Im nachfolgenden Abschnitt wird eine detaillierte Erläuterung der Anforderungen für optionale Aktivitäten und Kanten in GP-Modellen vorgenommen. (Bauer, 2025, S. 805)

Die Information, ob eine Aktivität optional ist, muss im GP-Template gespeichert werden können. Es ist essenziell, dass derartige Aktivitäten als optional gekennzeichnet werden. Diesbezüglich ist eine entsprechende Markierung innerhalb der Arbeitsliste der Endbenutzer erforderlich. Darüber hinaus ist es notwendig, dass im Rahmen der GP-Modellierung die Möglichkeit besteht, einen Text zu verfassen, der die Gründe und Zeitpunkte für das Überspringen von Aktivitäten beschreibt. Zudem ist zu gewährleisten, dass dieser Text von den Endnutzern einsehbar ist. Ein weiterer wesentlicher Aspekt ist die Festlegung der Personen, die befugt sind, die Aktivität zu überspringen. (Bauer, 2025, S. 805)

Darüber hinaus besteht die Möglichkeit, Regeln zu modellieren, die definieren, dass eine Aktivität automatisch übersprungen werden soll. Die Anwendung dieser Maßnahme erfolgt in Situationen, in denen eine Aktivität nicht länger erforderlich oder nützlich ist. Darüber hinaus soll die Möglichkeit bestehen, automatisierte Prozessschritte und ganze Teilprozesse als optional zu kennzeichnen. (Bauer, 2025, S. 805)

Die Nutzung optionaler Teilprozesse ermöglicht die Beeinflussung der "natürlichen" Arbeitsreihenfolge. Die beteiligten Personen haben die Möglichkeit, zu bestimmen, ob die

nachfolgende obligatorische Aktivität des Teilprozesses zeitgleich oder vorzeitig bearbeitet wird. (Bauer, 2025, S. 805)

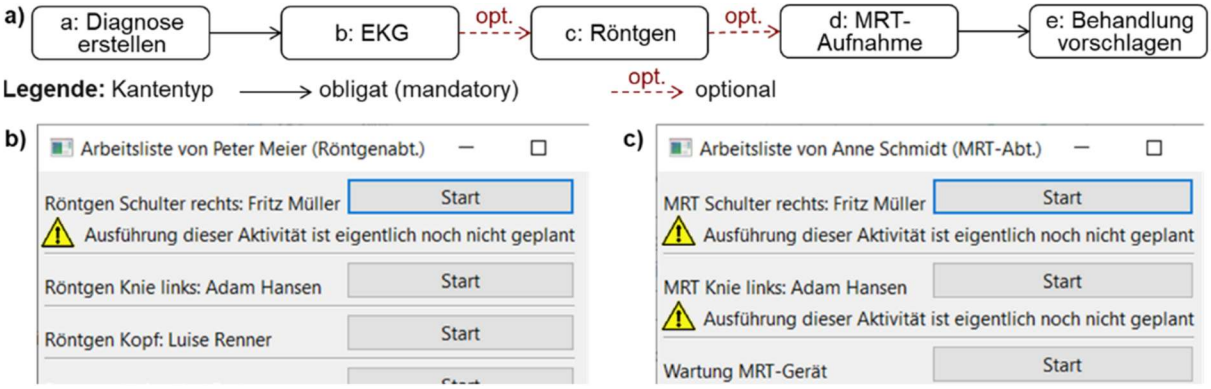

Abb 1 a) Untersuchungsprozess sowie b) Arbeitsliste für die Akt. c und c) für Akt. D (Bauer, 2023a, S. 4)

Die Darstellung einer gewünschten Ausführungsreihenfolge von Aktivitäten wird durch optionale Kanten ermöglicht. Es besteht jedoch keine zwingende Notwendigkeit, diese festgelegte Reihenfolge einzuhalten. Vielmehr können Aktivitäten je nach Bedarf frühzeitig initiiert werden. Zu diesem Zweck werden die Kennzeichnungen optional, mandatory und soft für Sequenzkanten eingeführt. Eine konventionelle Sequenzkante wird durch mandatory gekennzeichnet. Dies impliziert, dass die Aktivität a vor dem Start der Aktivität b beendet werden muss. Die Aktivitäten b und c sind mit optionalen Kanten verbunden, das gleiche gilt für die Aktivität c und d. Es sei darauf hingewiesen, dass die Fertigstellung von A den Start der Aktivitäten B, C oder D ermöglicht. Die Aktivitäten d und e sind mit einer mandatory Sequenzkante verbunden. Im Falle einer vorzeitigen Beendigung der Aufgabe d ist der Beginn der Aufgabe e erst nach Abschluss der Aufgaben b und c zulässig. Es sei die Hypothese aufgestellt, dass die Sequenzkante zwischen den Aktivitäten d und e die Kennzeichnung soft aufweist. Nach Abschluss der Arbeiten an Aktivität d könnte Aktivität e initiiert werden, ohne auf die Fertigstellung der Aktivitäten b und c warten zu müssen. Hierbei ist zu berücksichtigen, dass Sequenzkanten mit der Kennzeichnung soft ausschließnach nach Sequenzkanten mit Kennzeichnung optional verwendet werden dürfen. In der Realität wäre diese Annahme nicht sinnvoll, jedoch kann sie für das vorliegende Beispiel vernachlässigt werden. (Bauer, 2023a, S. 3-6)

Es lassen sich daraus die Anforderungen ableiten, dass Kanten optional, mandatory oder soft gekennzeichnet werden müssen. Darüber hinaus soll die Möglichkeit bestehen, zu einer Kante

einen Erklärungstext zu verfassen. Dieser hat den Zweck, die Bedingungen zu definieren, unter denen ein frühzeitiger Start als sinnvoll erachtet wird. (Bauer, 2025

, S. 805)

2.4.1.2 Spezielle Arten von Sequenzkanten

Bauer hat in diesem Zusammenhang vier verschiedene Typen von Sequenzkanten definiert.

EndBeforeStart repräsentiert die konventionelle Form der Sequenzkante, welche gemäß den Standards aller gängigen Modellierungssprachen verwendet wird. Sie bestimmt, dass die Aktivität A vor dem Start der Aktivität B beendet werden muss. Dieser Prozess resultiert in einer regulären Abfolge der Aktivitäten. (Bauer, 2022, S. 2)

StartBeforeStart befasst sich mit der ersten von Bauer beschriebenen erweiterten Kante. Gemäß ihrer Aussage kann die Aktivität C bereits nach dem Start von Aktivität B initiiert werden. Für die Ausführung dieser Prozedur ist es essenziell, dass Aktivität B gestartet, jedoch noch nicht vollständig beendet ist. Die vorliegende Konstellation ermöglicht eine parallele Bearbeitung von B und C. Die simultane Bearbeitung ermöglicht eine Reduktion der Entwicklungszeit und der Gesamtzeit des GP. (Bauer, 2022, S. 2)

EndBeforeEnd legt fest das Aktivität D erst nach der Fertigstellung von Aktivität C beendet werden darf. Diese Art von Sequenzkante findet Nutzen, sofern während der Entwicklung eines Steuergeräts bereits Tests durchgeführt werden sollen. Die Beendigung solcher Tests ist erst möglich, wenn das Steuergerät vollständig entwickelt ist. (Bauer, 2022, S. 2)

StartBeforeEnd legt fest das Aktivität E starten muss bevor Aktivität D beendet ist.

Eine weitere neu hinzugekommene Eigenschaft von Sequenzkanten ist die Ergänzung einer optionalen Zeitangabe. Diese Funktion ermöglicht die Festlegung einer Mindestwartezeit zwischen Aktivitäten oder einer maximalen Zeitbegrenzung für den gesamten GP. Des Weiteren besteht die Möglichkeit, eine automatisierte Warnungsnachricht zu modellieren, die im Falle einer Überschreitung der Maximalzeit ausgelöst wird. Es sei darauf hingewiesen, dass eine Kante eine minimale sowie eine maximale Zeit aufweisen kann. Durch die Erweiterung der Kanten ist es möglich komplexere Ablaufstrukturen einfacher und intuitiver zu gestalten. (Bauer, 2025, S. 807)

2.4.1.3Exklusiver Abschnitt

Das Segment des Exklusiven Abschnittes widmet sich der gegenseitigen Ausschließung. Sofern lediglich ein Prototyp zur Verfügung steht und zwei verschiedene Aktivitäten X und Y auf diesen angewiesen sind, ist es unmöglich, diese gleichzeitig zu bearbeiten. Es sei darauf hingewiesen, dass die beiden Aktivitäten aus unterschiedlichen Bereichen des GP stammen können. Ein solcher gegenseitiger Ausschluss kann durch einen sogenannten Exklusiven Abschnitt modelliert werden. Für diesen Zweck können willkürliche Aktivitäten einem oder mehreren Exklusiver Abschnitten zugewiesen werden. Es besteht die Möglichkeit, diese Zuweisungen als optional zu kennzeichnen. Dies kann vorteilhaft sein, sofern für spezifische Prozesse in Ausnahmefällen keine bestimmten Ressourcen benötigt werden. Für diesen Zweck wird eine Warnmeldung modelliert, die den Nutzern signalisiert, dass ein Start grundsätzlich möglich ist, jedoch nicht empfohlen wird. Darüber hinaus besteht die Möglichkeit, einen Erklärungstext zu verfassen, in dem die Gründe dargelegt werden, die die gleichzeitige Ausführung der Aktivitäten als unerwünscht erscheinen lassen. (Bauer, 2025, S. 807)

2.4.1.4 Dynamische Sprünge

Sollten Nutzer oder Nutzerinnen unerwartete Situationen während des GP bemerken, besteht die Möglichkeit, einen Sprung auszulösen. Der GP zeichnet sich durch einen dynamischen Wechsel der Aktivitäten aus. Zu diesem Zweck werden eine valide Startaktivität, Zielaktivität, berechtigte Nutzer und Nutzerinnen sowie das beabsichtigte Verhalten der Aktivitäten vormodelliert. Die Ausführung von Sprüngen kann zu beliebigen Zeitpunkten initiiert werden. (Bauer 2024a) Sprünge können in drei verschiedenen Richtungen erfolgen. Die grundlegenden Richtungen, die in diesem Kontext von Relevanz sind, sind vorwärts, rückwärts und seitwärts. In diesem Kontext ist zu berücksichtigen, dass vorwärts Sprünge genutzt werden können, um nicht obligatorische Aktivitäten zu überspringen, falls die geplante Prozesszeit sich dem Ende nähert. Darüber hinaus besteht die Möglichkeit, Aktivitäten zu überspringen, sofern diese im Laufe des Prozesses obsolet geworden sind, sei es aufgrund neuer Erkenntnisse oder gesetzlicher Änderungen. Zudem besteht die Möglichkeit, mittels rückwärts Sprüngen eine Wiederholung spezifischer Aktivitäten zu initialisieren. Dieser Vorgang erweist sich als vorteilhaft, wenn fehlerhafte Daten verwendet wurden oder Aktivitäten fehlerhaft ausgeführt wurden. Die Ausführung von Sprüngen seitwärts ermöglicht die Bewegung innerhalb und außerhalb des betreffenden Bereichs. Die simultane Durchführung von Aktivitäten in diversen Regionen ist eine Möglichkeit. (Bauer, 2025, S. 807)

Bauer hat eine Reihe von Anforderungen definiert, die dynamischen Sprüngen zugrunde liegen. Zunächst ist festzuhalten, dass für jeden Sprung mehrere Start- und Zielaktivitäten modelliert werden müssen. Im Falle des Auslösens eines Sprungs während der Ausführungszeit des GP ist es erforderlich, dass ein autorisierter Nutzer oder eine autorisierte Nutzerin eine spezifische Zielaktivität pro parallelen Zweig auswählen kann. Darüber hinaus sind Startaktivitäten unabhängig voneinander, da zwischen ihnen andere Aktivitäten existieren können und ein Sprung während ihrer Ausführung wenig sinnvoll ist. Darüber hinaus können Zwischenaktivitäten auftreten, bei denen es aus pragmatischen Gründen nicht sinnvoll ist, die Aktivität nach dem Sprung fortzusetzen. (Bauer, 2025, S. 807)

Zudem besteht die Option, Aktivitäten als Standardziel für spezifische Sprünge zu definieren. Darüber hinaus soll jedem Zweig die Möglichkeit gegeben werden, eine Aktivität als Standardziel zu definieren. Diese Vorgehensweise trägt zur Effizienzsteigerung bei der Selektion von Zielen für Sprünge bei. Selbstverständlich besteht für Nutzerinnen und Nutzer die Option, alternative vordefinierte Ziele auszuwählen. Eine Aktivität kann für mehrere Sprünge als Start- oder Zielaktivität definiert werden. (Bauer, 2025, S. 807-808)

In Bezug auf den Kontrollfluss kann ein Sprung verschiedene Richtungen annehmen. Es besteht die Möglichkeit, einen Sprung vorwärts, rückwärts oder seitwärts in einen anderen XOR- oder OR-Zweig eines GP auszuführen. Infolgedessen werden Sprünge in und aus Bereichen, in denen gleichzeitig Aktivitäten ausgeführt werden, ermöglicht. (Bauer, 2025, S. 808)

Für jede Aktivität sollte eine gewünschte Verhaltensweise beim Sprung ausgewählt werden können. Für diesen Zweck werden vormodellierte Konfigurationsoptionen bereitgestellt. Es besteht die Möglichkeit, diese für Start- und Zielaktivitäten sowie für die Aktivitäten zwischen und nach ihnen zu definieren. Bauer beschreibt für diesen Sachverhalt drei Konfigurationsoptionen, die als CatchUpMode, RepeatMode und ContinueMode bezeichnet werden. (Bauer, 2025, S. 808)

CatchUpMode ist relevant für Vorwärtssprünge. Diese Konfiguration definiert, ob übersprungene Aktivitäten nachgeholt werden sollen. Dies ist von Nutzen, sofern der Output der Aktivitäten zu einem späteren Zeitpunkt benötigt wird. (Bauer, 2025, S. 808)

In Bezug auf den Aspekt der Rückwärtssprünge erweist sich die Funktion RepeatMode als relevant. Die vorliegende Konfiguration spezifiziert, ob Aktivitäten nach dem Sprung erneut ausgeführt werden sollen. Diese Feststellung ist von signifikanter Relevanz, da eine

Modifikation des Outputs der Aktivitäten nicht auszuschließen ist. Für diesen Zweck erfolgt eine weitere Unterteilung des RepeatMode in drei verschiedene Varianten. Im RepeatMode(X)=Discard werden die Resultate der ursprünglichen Ausführung gelöscht und die Aktivität wird erneut ausgeführt. Darüber hinaus besteht die Option, den RepeatMode(X)=Control zu nutzen. In diesem Prozess werden die ursprünglichen Resultate durch die beteiligten Akteure geprüft. Ziel dieses Prozesses ist es festzustellen, ob eine Anpassung der Resultate erforderlich ist. Im Anschluss erfolgt ein erneuter Start der Aktivität unter Verwendung vorausgefüllter Quelldaten. Des Weiteren gibt es den RepeatMode(X)=Keep, bei dem die ursprünglichen Daten beibehalten werden und die Aktivität nicht wiederholt wird. (Bauer, 2025, S. 808)

2.4.2 Entwicklung des Conceptual Domain Model of the Extension

Hierfür hat Bauer die Stroppi-Methode sowie die Braun-Methode angewandt. Gemäß der Stroppi-Methode wurde seitens Bauers zunächst das UML-Klassen-Diagramm CDME erstellt. Das vorliegende Model umfasst sämtliche Klassen, Assoziationen und Attribute, welche erforderlich sind, um das zuvor beschriebene Modell zu implementieren. Die Braun-Methode wurde angewendet, um zu gewährleisten, dass neue Klassen ausschließlich bei Bedarf erstellt werden. In diesem Kontext erfolgten eine Identifikation und Diskussion der geeignetsten Klassen für die Erweiterung, sowie eine Analyse der für ihre Verwirklichung relevanten BPMN-Standards. (Bauer, 2025, S. 808)

Abb. CDME

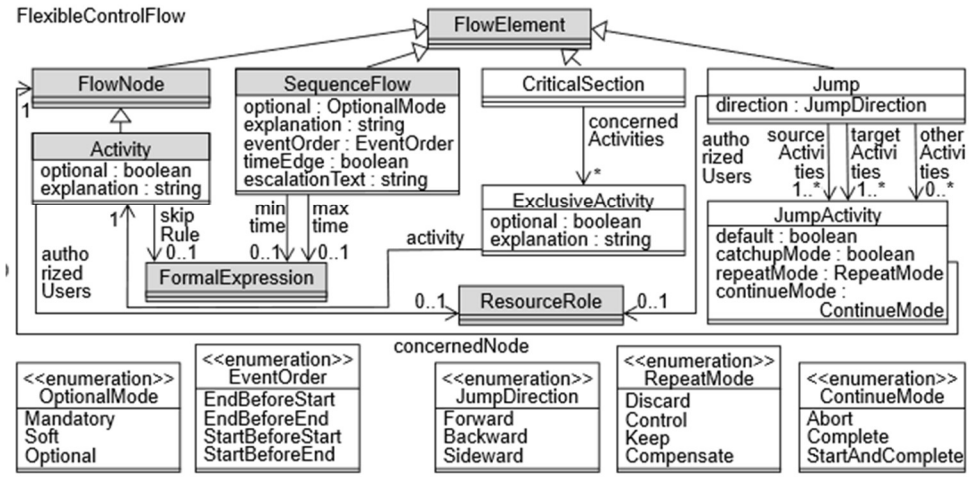

Abb. 2 CDME for the Extensions Required to Pre-Model Flexibility. (Bauer, 2025)

2.4.2.1 Optionale Aktivitäten und Kanten

Für die Realisierung von Optional Activitites wird das BPMN-Element Activity erweitert. In Anbetracht der Tatschache, dass User Task und sogar ganze Teilprozesse optional sein können, erweist sich das BPMN-Element Task als nicht geeignet. Das BPMN-Element FlowNode ist als zu allgemein zu betrachten, da es nicht überspringbare Gateways beinhaltet. (Bauer, 2025, S. 808)

Das Element Activity wird durch die Atttribute optional und explanation sowie die Assoziationen authorizedUsers und skipRule erweitert. Diese Erweiterung wurde auf Grundlage der zuvor definierten Anforderungen festgelegt. Das Attribut optional ist dem Datentyp boolean zuzuordnen und dient der Kennzeichnung von optionalen Aktivitäten. Für die Endnutzer wird eine Erklärung in Form eines Strings unter explanation gespeichert. Die variable authorizedUsers assoziiert eine Activity mit einem BPMN-Element von Typ ResourceRole. Es sei darauf hingewiesen, dass diese Assoziation nicht obligatorisch ist und bei der Modellierung weggelassen werden kann. Die vorliegende Implementation gestattet eine Kardinalität von 0, was zur Konsequenz hat, dass alle Personen, die von der Activity betroffen sind, über die Berechtigung verfügen, die Activity zu überspringen. Die Assoziation skipRule wird für die Zueweisung einer FormalExpression verwendet. Wird das boolean true zurückgegeben, wird die Activity automatisch ausgelassen. Die vorliegende Assoziation kann weggelassen werden, wodurch ein manuelles Anstoßen des Überspringens erforderlich wird. (Bauer, 2025, S. 808-809)

Für die Realisierung von Optional Edges erfolgt eine Erweiterung des BPMN-Elements Sequence-Flow. Zu diesem Zweck wird das Element mit den Attributen optional und explanation erweitert Das Attribut optional dient der Beschreibung der Art der Kante. Im Gegensatz zum boolean von Optional Activity wird hier eine Enumeration namens optionalMode verwendet. Sie umfasst die Werte Mandatory, Soft oder Optional.

Die Eigenschaft explanation speichert eine Erklärung als String, wenn die frühzeitige Ausführung der Zielaktivität sinnvoll ist. (Bauer, 2025, S. 808-809)

2.4.2.2 Spezielle Arten von Sequenzkanten

Die speziellen Arten von Sequenzkanten stellen auch eine Erweiterung des Elements SequenceFlow dar. In der Konsequenz dessen werden die Attribute eventOrder, timeEdge und excalationText sowie die Assoziationen minTime und maxTime dem SequenceFlow hinzugefügt. (Bauer, 2025, S. 808-809)

Das Element eventOrder dient der expliziten Bezeichnung, ob der Start- oder das Endevent einer Aktivität referenziert wird. Zu diesem Zweck wird eine Enumeration mit der Bezeichnung EventOrder generiert. Die Enumeration umfasst die vier von Bauer beschriebenen Arten spezieller Kanten. Die EndBeforeStart-, EndBeforeEnd-, StartBeforeStart- und StartBeforeEnd-Sequenzkanten. (Bauer, 2025, S. 808-809)

Das boolean timeEdge zeigt an ob es sich um eine Zeitkante handelt. Die Assoziationen minTime und maxTime assozieren das Element SequenceFlow mit dem BPMN-Element FormalExpression. Ihre primäre Funktion besteht in der Zeitberechnung. Für diesen Zweck kann eine einfache Zeitangabe genutzt werden, die beispielsweise zur Endzeit einer vorangehenden Aktivität addiert wird, um eine frühe oder spätmöglichste Startzeit zu ermitteln. Des Weiteren besteht die Möglichkeit, komplexere Zeitangaben zu spezifizieren, wie beispielsweise eine exakte Uhrzeit an einem spezifischen Tag. Es sei darauf hingewiesen, dass beide Assoziationen die Kardinalität 0 aufweisen und daher weggelassen werden können. Infolgedessen können die früheste und späteste Startzeit willkürlich gewählt werden. Das Attribut escalationText repräsentiert eine Warnnachricht für die Nutzer, die in Form eines Strings gespeichert wird. (Bauer, 2025, S. 808-809)

2.4.2.3 Exklusier Ausschluss

Bauer hat festgestellt, dass BPMN über kein geeignetes Standardelement für die Critical Section verfügt, das für diese Erweiterung genutzt werden kann. Gemäß Bauer wird die Critical Section folglich als eine Spezifizierung des FlowElements dargestellt. Diese Schlussfolgerung basiert auf der Tatsache, dass CriticalSection die Ausführungsreihenfolge von Activities beeinflusst. Für jede betroffene Activity werden mehrere Attributwerte modelliert. Da diese Werte sich unterscheiden können für jede Activity einer CriticalSection, führt Bauer den Elementtypen ExclusiveActivity ein. Dieser Elementtyp findet Anwendung bei der Generierung separater Elemente desselben Typs für jedes betroffene Activity-Element und der Speicherung der zugehörigen Attributwerte. Die Assoziation concernedActivities ordnet mehrere Elemente von Typ ExcluseActivity einer CriticalSection zu. Das Element ExclusiveActivity besteht aus den Attributen optional und explanation sowie die Assoziation activity. Das Attribut optional zeigt an ob die Zuweising optional ist durch die Verwendung eines boolean. Das Attribut explanation ist ein Erklärungstext in Form eines Strings. Die Assoziation activity fungiert als Bindeglied zwischen einer ExclusiveActivity und einer Activity. Es besteht demnach die Möglichkeit, eine Activity an mehrere Instanzen von

CriticalSection zu binden. Die Assoziation einer Activity an mehrere Instanzen von ExclusiveActivity ist dabei eine notwendige Voraussetzung. (Bauer, 2025, S. 809-810)

2.4.2.4 Dynamische Sprünge

Bauer hat eine Spezialisierung eines FlowElements namens Jump eingeführt, wodurch die Realisierung dynamischer Sprünge ermöglicht wird. Der Elementtyp FlowElement wird hier verwendet da Sprünge die Ausführungsreihenfolge von Activity beeinflussen. Des Weiteren wird Jump zur Definition von vormodellierten Sprüngen verwendet. Hierfür wird Jump mit dem Attributen direction und den Assoziationen sourceActivities, targetActivitites, otherActivitites und autherizedUsers ausgestattet.

Das Attribut direction spezifizert die Richtung des Sprunges. Aus diesem Grund wird die Enumeration JumpDirection eingeführt, welche die Werte Forward, Backward und Sideward umfasst. Die Richtung des Sprungs hat signifikante Auswirkungen auf das Verhalten des Sprungs.

Die Assoziation authorizedUsers referenziert eine ResourceRole, um die spezifischen Personen zu definieren, die dazu befugt sind, diesen Sprung auszulösen.

Die Assoziationen sourceActivities, targetActivitites und otherActivitites referenzieren mehrere Instanzen eines weiteren neuen Elements namens JumpActivity. JumpActivity ist ein Element des Typs FlowNodes, da Sprünge die Fähigkeit besitzen, jegliche Arten von Activity und Gateways zu beeinflussen. Darüber hinaus besteht die Möglichkeit, Sprünge unmittelbar nach einem Gateway zu starten oder ein Gateway als Ziel zu definieren. Es ist eine zwingende Voraussetzung, dass die Assoziationen sourceActivities und targetActivities mindestens eine JumpActivity referenzieren. Andernfalls bleiben sowohl der Start als auch das Ziel eines Sprungs unbestimmt. Die Assoziation otherActivites kann genutzt werden, um Aktivitäten, die sich zwischen der Start- und Zielaktivität befinden, mit Konfigurationsoptionen zu versehen.

Die zuvor genannte JumpActivity setzt sich aus den Attributen default, catchupMode, repeatMode und continueMode und der Assoziations concernedNode zusammen.

Die Komponente default dient der Kennzeichnung, ob die Aktivität als Standardziel fungieren soll. Für diesen Schritt wird lediglich ein boolean verwendet.

Die Attribute catchupMode, repeatMode und continueMode realisieren die zuvor beschriebenen Konfigurationsoptionen eines Sprunges. Sie sind Bestandteil einer JumpActivity, da die Werte für unterschiedliche Aktivitäten desselben Sprungs variieren

können. In Bezug auf den für catchupMode verwendeten Boolean ist anzumerken, dass lediglich zwei Werte möglich sind, nämlich die Frage, ob die Aktivität nachgeholt werden soll oder nicht. Für die Variablen repeatMode und continueMode werden Enumerationen verwendet und angemessene Werte werden definiert.

2.4.3 Ableitung des BPMN Plus Extension Model

„Durch Anwendung der Stroppi-Methodik wird aus dem in Abb2. dargestellte CDME das in Abb. 3 dargestellte BPMN+X-Modell FlexibleControlFlow mit dem Stereotyp «ExtensionModel» abgeleitet. BPMN-Kernklassen werden dabei als «BPMNElement» übernommen. Zusätzliche Attribute zu bestehenden BPMN-Klassen werden über eigene Klassen mit dem Stereotyp «ExtensionDefinition» eingeführt. Neue CDME-Klassen, die direkte Untertypen von BPMN-Klassen sind, werden ebenfalls als «ExtensionDefinition» modelliert. Neue Klassen, die ausschließlich Beziehungen zu BPMN-Klassen aufweisen, erhalten den Stereotyp «ExtensionElement». Aufzählungstypen werden als «ExtensionEnum» geführt. Generalisierungen zwischen originalen BPMN-Elementen bleiben erhalten. Generalisierungen von neuen zu BPMN-Klassen werden in Assoziationen mit dem Stereotyp «ExtensionRelationship» überführt. Assoziationen neuer Klassen werden sowohl zwischen neuen Klassen als auch zwischen neuen und BPMN-Klassen übernommen.“ (Bauer, 2025, S. 810)

Abb. BPMN+X

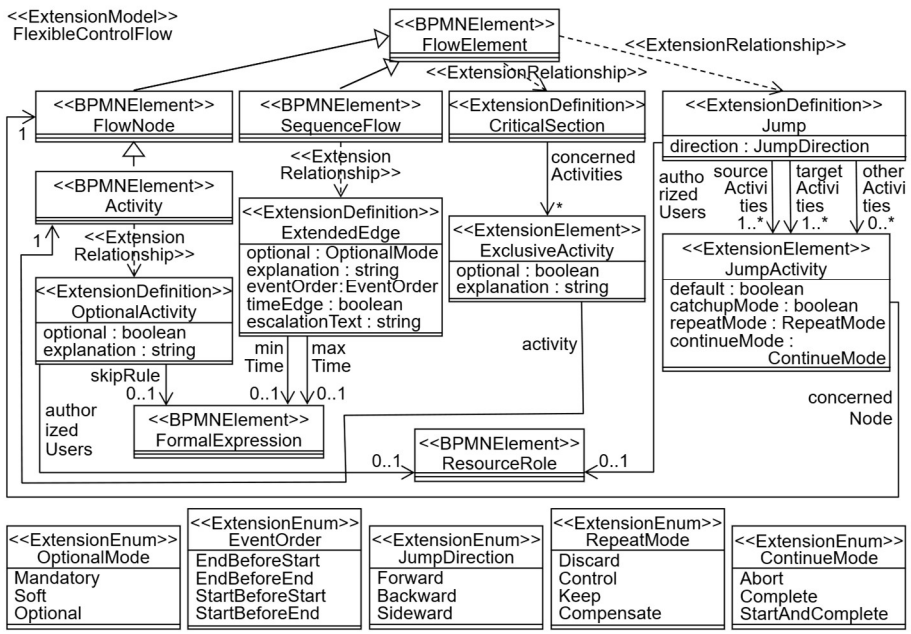

Abb 3 BPMN Plus Extension (BPMN+X) Model (Bauer, 2025, S. 810)

2.4.4 Überführung in ein XML-Schema Extension Model

„Bauer überführt das BPMN+X-Modell FlexibleControlFlow (vgl. Abb. 3) in ein XML-Schema (vgl. Abb. 4) durch die Verwendung der Stroppi-Methodik. Jedes Element mit dem Stereotyp «ExtensionDefinition» wird in eine «GroupDefinition» transformiert, die für jedes Attribut und für jede Assoziation eine «ElementDeclaration» enthält. Elementare Datentypen werden unverändert übernommen. Assoziationen zu BPMN-Elementen werden über Felder des Typs QName als Referenzen auf die entsprechenden Modellelemente abgebildet. Elemente mit dem Stereotyp «ExtensionElement» werden als «ComplexTypeDefinition» modelliert und durch ein vorangestelltes **t** als Datentyp gekennzeichnet; die zugehörigen «ElementDeclaration»-Einträge entstehen nach demselben Muster wie bei «ExtensionDefinition». Aufzählungstypen mit dem Stereotyp «ExtensionEnum» gehen in eine «SimpleTypeDefinition» mit ihren zulässigen Ausprägungen über. Bereits im BPMN-Standard definierte «BPMNElement»-Klassen werden nicht erneut im Erweiterungs-Schema definiert, sondern ausschließlich über QName-Referenzen verwendet.“ (Bauer, 2023b, S.12)

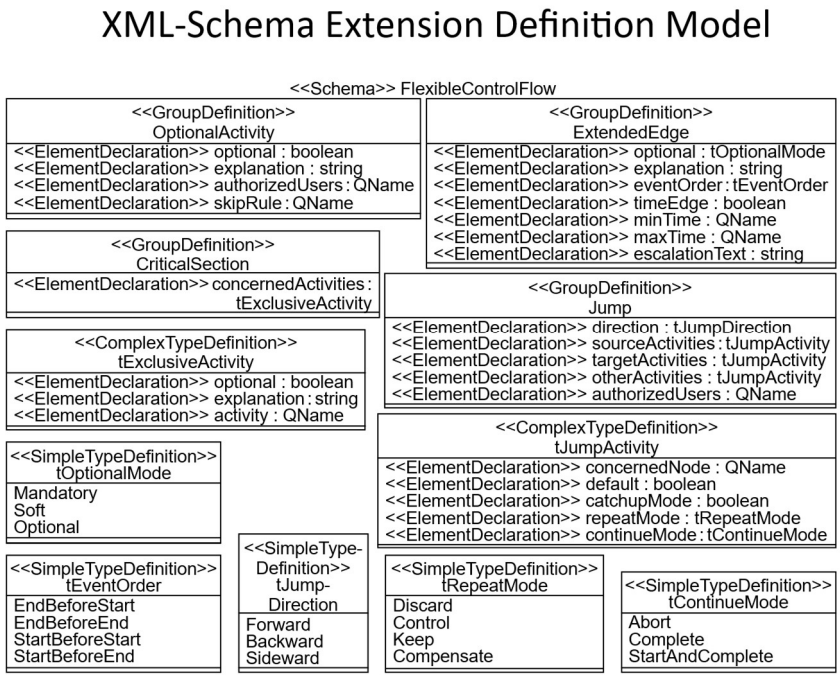

Abb 4 XML-Schema Extension Definition Model nach Stroppi (Bauer, 2023b, S.12)

3. Toolauswahl

Im weiteren Verlauf werden diverse BPMN-Tools präsentiert, die im Rahmen dieser Arbeit einer Analyse unterzogen wurden.

BPMN+X

In der Stroppi-Methode wird die Verwendung des BPMN+X Tools empfohlen. Das Tool ist ein Plugin für die Integrierte Entwicklungsumgebung Eclipse der Eclipse Foundation. Das vorliegende Plugin basiert auf den UML2 und UML2 Tools Plugins. Es ist bedauerlicherweise zu vermerken, dass das Tool seit 2012 keiner weiteren Entwicklung unterzogen wurde. In der Folge war eine Installation des Plugins nicht möglich. Die BPMN+X-Erweiterung ist abhängig von Eclipse-Packages, welche nicht mehr zur Verfügung stehen. Darüber hinaus war es mir nicht möglich, die erforderlichen UML2- und UML2-Tool-Plugins zu installieren, da diese auch von älteren Paketen abhängig sind. Der vorliegende Vorgang wurde zudem mit älteren Versionen von Eclipse Indigo, wie auf der Website angegeben, durchgeführt. Jedoch konnte dieses Vorgehen nicht erfolgreich abgeschlossen werden. Darüber hinaus ist die Funktionalität der Website eingeschränkt. Einige Verknüpfungen und Einträge führen zu keinen Ergebnissen. Zunächst wird ein Extension Model mit dem Namen der Erweiterung erstellt. Des Weiteren wurden von Stroppi zwei weitere Tools vorgeschlagen: das BPMN+X To XML Schema M2M Transformation-Plug-IN und das XML Schema M2C Transformation. Nach einer umfassenden Recherche konnten keine der beiden genannten Tools ausfindig gemacht werden.

Signavio

Signavio ist eine webbasierte GP-Modellierungsplattform von SAP. Im Rahmen meines Studiums erhielt ich eine akademische Lizenz für die Plattform. Die Plattform bietet benutzerdefinierte Attribute und Grafiken. Somit wäre sie rein theoretisch erweiterbar. Auf diese Funktionen hatte ich aufgrund meiner akademischen Lizenz leider keinen Zugriff.

Bee-Up

Bee-Up ist ein Modellierungstool der OMiLAB Organization, das auf ADOxx basiert. Das Tool wird hauptsächlich als Lernwerkzeug genutzt und unterstützt die Modellierungssprachen BPMN, Event-Driven Process Chains, Entity-Relationship-Diagramme, UML und Petri-Netze. Es kann durch Add-ons erweitert werden. Dadurch können nutzerdefinierte Skripte,

Webserviceaufrufe oder Systemanwendungen ausgeführt werden. Dieses Add-on-System kann jedoch nicht die durch Bee-Up verfügbaren Modelle, Objekte, Beziehungen und Attribute beeinflussen, weshalb es für meine Zwecke unbrauchbar ist. S.24

Adonis

Adonis ist eine webbasierte GP-Modellierungsplattform der BOC Group. Sowohl das Tool selbst als auch das Webframework AdonisJS sind erweiterbar. Adonis lässt sich durch zusätzliche Module und Funktionen ergänzen, während AdonisJS auf einem modularen Framework basiert. Leider ist für den Zugriff auf diese Funktionen eine teure Lizenz erforderlich, weshalb ich mich gegen dieses Tool entschieden habe. Da ohne vorheriges Testen nicht gewährleistet werden kann, dass dieses Tool für meine Zwecke geeignet ist,

Open-BPMN

Auf der Webseite https://www.open-bpmn.org/ befindet sich eine Verlinkung, die es ermöglicht, die Plattform als Online-Tool über den Internetbrowser zu testen. Dieser Versuch misslingt und dem Benutzer wird eine Fehlermeldung angezeigt. In der Folge wurde eine detaillierte Analyse des offenen Repositorys auf GitHub durchgeführt. Die Konzeption der Plattform beinhaltet die Option für die Benutzer, eine individuelle Erweiterung zu entwickeln. Aus diesem Grund wurde die initiale Vorgehensweise bestimmt, die Entwicklungsumgebung für das Projekt auf einem privaten Windows-Rechner zu implementieren. Dieser Aspekt war in der ursprünglichen Programmierung nicht berücksichtigt worden. Im Rahmen des Prozesses manifestierten sich zahlreiche Hindernisse, die vorab nicht antizipiert wurden. Die Zielanwendung erfordert jedoch eine komplexe Kombination mehrerer Tools und Frameworks, die primär auf Unix-basierten Systemen optimiert sind. Der Open-BPMN ist eine freie und offene Modellierungsplattform. Die vorliegende Software wird gegenwärtig noch im Rahmen eines Open-Source-Projekts weiterentwickelt und ist gemäß ihrer Beschreibung erweiterbar. Die Projektumgebung basiert auf einer Reihe von Technologien, die in einem modernen Entwicklungsstack typisch sind. Bereits während der initialen Einrichtung wurde deutlich, dass die Windows-Umgebung nur eingeschränkt mit den projektspezifischen Anforderungen kompatibel war. Um diese Einschränkungen zu umgehen, wurde der Versuch unternommen, das Windows Subsystem for Linux (WSL) einzurichten, um eine Linux-ähnliche Umgebung bereitzustellen. Die Installation von WSL gestaltete sich jedoch als problematisch. Auf dem verwendeten System kam es wiederholt zu Fehlern während der Aktivierung der Virtualisierungsfunktionen (Hyper-V, Virtual Machine Platform).

Diese sind für den Betrieb von WSL zwingend erforderlich, wurden jedoch vom System aufgrund inkompatibler BIOS-Einstellungen oder einer bereits aktiven Drittanbieter-Virtualisierung (VMware) blockiert. Obwohl die Teilinstallation von WSL erfolgreich war, traten bei der Integration mit Docker Desktop weitere Schwierigkeiten auf. Für die Ausführung von Linux-basierten Containern ist eine korrekte Kommunikation mit der WSL-Engine erforderlich. Diese Kommunikation konnte nicht erfolgreich durchgeführt werden. Ein weiterer Engpass manifestierte sich in der Verwendung von Bash-Skripten im Projekt. Diese finden unter anderem zur Automatisierung im Entwicklungsprozess Anwendung. Die Ausführung dieser Skripte war unter Windows aufgrund einer Limitierung der Standard-Windows-Shell/Powershell nur in eingeschränktem Maße möglich. Zusammenfassend lässt sich feststellen, dass die Einrichtung einer produktiven und vollständigen Entwicklungsumgebung unter Windows als nicht praktikabel zu erachten ist. Eine Analyse der vorliegenden Problematik ergibt, dass eine Kombination aus Virtualisierungsproblemen, Docker- und WSL-Kompatibilitätsfehlern, Dateisystemunterschieden sowie der fehlenden nativen Unterstützung für Unix-Tools die lokale Ausführung erschwert. In Anbetracht dieser Einschränkungen wurde der Ansatz, die Entwicklungsumgebung auf dem persönlichen Windows-Rechner zu betreiben, verworfen. Stattdessen wurde die Entscheidung getroffen, die Umgebung auf einem dedizierten System mit einem nativen Linux-Betriebssystem zu installieren. Die vorliegende Umstellung resultierte in einer signifikanten Reduktion des Konfigurationsaufwands. Es wurde eine robuste und konsistente Grundlage für die Erweiterungsentwicklung von Open BPMN geschaffen.

4. Erstellung einer XSD-Datei für die BPMN-Erweiterung

Im Rahmen dieser Arbeit wurde mithilfe der Stroppi-Methode aus dem von Bauer erstellten XML-Schema-Extension-Definition-Modell ein XML-Schema-Dokument generiert. (vgl. Abb. 5., das komplette XSD ist im Anhang einsehbar)

Zu diesem Zweck wird eine Transformation des <<schema>> FlexibleControllFlow in das <xs:schema targetNamespace="FlexibleControlFlow.xsd"> durchgeführt. Die Stereotypen <<ElementDeclartion>> werden in <xs:element> umgewandelt und adaptiert. Hierfür wird das neue Element benannt und ihm wird ein zulässiger Datentyp zugewiesen. Darüber hinaus werden verlorengegangene Assoziationen wieder eingeführt. Im XML-Schema-Extension-Definition-Modell ist es beispielsweise nicht länger möglich, zu erkennen, ob ein Jump keine oder mehrere otherActivities aufweisen kann. Die Repräsentation dieser Sachverhalte erfolgt durch die Implementierung der Indikatoren minOccur und maxOccur.

<xs:element name="otherActivities" type="tJumpActivity" minOccurs="0" maxOccurs="unbounded"/>

Des Weiteren werden die Stereotypen <<GroupDefinition>> werden in ein <xs:group> umgewandelt, <<ComplexTypeDefinition>> werden in <xs:complexType> transformiert und die Stereotypen <<SimpleTypeDefinition>> werden in <xs:simpleType> umgewandelt.

```
<xs:group name="Jump">
    <xs:sequence>
        <xs:element name="direction" type="tJumpDirection"/>
        <xs:element name="sourceActivities" type="tJumpActivity" maxOccurs="unbounded"/>
        <xs:element name="targetActivities" type="tJumpActivity" maxOccurs="unbounded"/>
        <xs:element name="otherActivities" type="tJumpActivity" minOccurs="0" maxOccurs=
        "unbounded"/>
        <xs:element name="authorizedUsers" type="tAuthorizedUser" minOccurs="0"/>
    </xs:sequence>
</xs:group>
<xs:element name="tJumpDirection">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:enumeration value="Forward"/>
            <xs:enumeration value="Backward"/>
            <xs:enumeration value="Sideward"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

Abb 5Beispiel für XSD-Umsetzung

5. Entwicklung einer BPMN-Erweiterung

Schwierigkeiten in der Entwicklungsphase

Während der Implementationsphase traten mehrere unvorhergesehene technische Herausforderungen auf. Das vorrangige Ziel bestand darin, eine Erweiterung für ein bereits existierendes Programm zu entwickeln. Nach einer Vielzahl von Versuchen wurde festgestellt, dass das Vorhaben nicht realisierbar ist. Infolge des fortgeschrittenen Alters der Programme manifestierten sich Schwierigkeiten bezüglich der Software-Kompatibilität. Es konnte festgestellt werden, dass ein signifikanter Anteil der im Programm eingesetzten Technologien entweder keine aktuellen Versionen aufweisen oder keine vergleichbaren Alternativen zur Verfügung stehen. Im Laufe der Zeit hat sich der Fokus meiner Arbeit auf die Recherche nach alternativen Frameworks oder Tools verlagert. Nach extensiver Suche wurde das Open-BPMN repository von der imxis als Erweiterungziel ausgewählt.

Bei der Sotware handels es sich um eine in aktiver Entwicklung befindenen Frameworks. Es modeliert einene BPMN 2.0 Standard mit der Verwendung der Eclipse Graphical Language Server Platform (GLSP) Architektur. Es besteht aus drei Basismodulen:

- open-bpmn.metamodel
- open-bpmn.glsp-server
- open-bpmn.glsp-client

Obwohl es sich bei der Entdeckung des Open-BPMN Projektes um eine vielversprechende Lösung für die Lücke der BPMN Modelierungs Software handelte, beinhaltete es viele Schwierigkeiten. Das vorliegende Repository ist eine Ansammlung von Code-Bestandteilen, die in mehreren Subprojekten organisiert sind.

Die Dokumentation für das Projekt auf open-bpmn.org war auch an einigen Stellen bereits veraltet. Da sich das Framework in aktiver Entwicklung befindet, sind einige Funktionsnamen refactored worden. Beispielsweise die Funktion configureBPMNExtensions in der Klasse BPMNDiagramModule wurde zu configureBPMNElementExtensions umgeschrieben.

6. Fazit

Die Identifizierung eines adäquaten BPMN-Tools stellte eine signifikante Herausforderung dar. Eine Analyse des Angebots an Erweiterbaren BPMN-Tools ergibt, dass viele dieser Tools entweder schwer zugänglich sind oder sehr veraltet. Es konnte festgestellt werden, dass ein signifikanter Anteil der Dokumentationen nicht mehr dem neusten Stand sind und sind daher keine ausreichende Grundlage für die Entwicklung von Leien bietet. Aufgrund der hohen Komplexität und der fehlenden Vertrautheit mit der Codebasis wurde ein signifikanter Anteil der Arbeitszeit für die Auseinandersetzung mit dem Projekt aufgewendet. Eine weitere Problematik, die zu einer Verzögerung der Entwicklung geführt hat, ist mein unzureichender Kenntnisstand. Es besteht die Möglichkeit, eine BPMN-Erweiterung mit dem open-bpmn-Tool zu entwickeln, auch wenn dies in meinem Fall nicht zum Erfolg geführt hat. Mir ist es lediglich gelungen die simplen Datenstrukturen aus dem UML-Diagramm umzusetzen. Ein Fokus zukünftiger Arbeiten sollte daher auf der Entwicklung oder Erweiterung von BPMN-Editoren liegen, die eine flexible Anpassung des Metamodells erlauben und so die Realisierung domänenspezifischer Erweiterungen erleichtern.

Literaturverzeichnis

Bauer T. (2022) *Verhalten und Ausführungssemantik von erweiterten Sequenzkanten in Geschäftsprozessen*

Bauer T. (2023a) *Verhalten und Ausführungssemantik optionaler Kanten in Geschäftsprozessen*

Bauer T. (2025) Extending *BPMN to Enable the Pre-Modelling of Flexibility for the Control Flow of Business Processesfor the Control Flow of Business Processes*

Bauer T. (2024) *A Formal Execution Semantics for Sophisticated Dynamic Jumps Within Business Processes*

OMG, (2011) *Object Management Group - Business Process Model and Notation (BPMN) 2.0.2.*

Bauer T. (2023b), *BPMN-Erweiterungen zur Vormodellierung von Flexibilität für den Kontrollfluss von Geschäftsprozessen"*

OMG, (2017) *Unified Modeling Language, v2.5.1*

Stroppi, L.J.R., Chiotti, O., Villarreal, P.D., 2011. *Extending BPMN 2.0: Method and Tool Support*

Abbildungsverzeichnis

Anhang

FlexibleControlFlow.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="FlexibleControlFlow.xsd"
xmlns:bpmn="http://www.omg.org/spec/BPMN/20100524/MODEL" elementFormDefault="qualified">
	<xs:import namespace="http://www.omg.org/spec/BPMN/20100524/MODEL" schemaLocation="BPMN20.xsd"/>
	<xs:element name="tAuthorizedUser">
		<xs:complexType>
			<xs:sequence>
				<xs:element ref="bpmn:ResourceRole"/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
	<xs:element name="tJumpDirection">
		<xs:simpleType>
			<xs:restriction base="xs:string">
				<xs:enumeration value="Forward"/>
				<xs:enumeration value="Backward"/>
				<xs:enumeration value="Sideward"/>
			</xs:restriction>
		</xs:simpleType>
```

```
</xs:element>
<xs:element name="tRepeatMode">
	<xs:simpleType>
		<xs:restriction base="xs:string">
			<xs:enumeration value="Discard"/>
			<xs:enumeration value="Control"/>
			<xs:enumeration value="Keep"/>
			<xs:enumeration value="Compensate"/>
		</xs:restriction>
	</xs:simpleType>
</xs:element>
<xs:element name="tContinueMode">
	<xs:simpleType>
		<xs:restriction base="xs:string">
			<xs:enumeration value="Abort"/>
			<xs:enumeration value="Complete"/>
			<xs:enumeration value="StartAndComplete"/>
		</xs:restriction>
	</xs:simpleType>
</xs:element>
<xs:element name="tOptionalMode">
	<xs:simpleType>
		<xs:restriction base="xs:string">
			<xs:enumeration value="Mandatory"/>
			<xs:enumeration value="Soft"/>
			<xs:enumeration value="Optional"/>
		</xs:restriction>
	</xs:simpleType>
</xs:element>
<xs:element name="tEventOrder">
	<xs:simpleType>
		<xs:restriction base="xs:string">
```

```
                    <xs:enumeration value="EndBeforeStart"/>
                    <xs:enumeration value="EndBeforeEnd"/>
                    <xs:enumeration value="StartBeforeStart"/>
                    <xs:enumeration value="StartBeforeEnd"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:element name="tJumpActivity">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="concernedActivites" type="QName"/>
                    <xs:element name="default" type="xs:boolean"/>
                    <xs:element name="catchupMode" type="xs:boolean"/>
                    <xs:element name="repeatMode" type="tRepeatMode"
minOccurs="0"/>
                    <xs:element name="continueMode" type="tContinueMode"
minOccurs="0"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:group name="OptionalAktivity">
            <xs:sequence>
                <xs:element name="optional" type="xs:boolean"/>
                <xs:element name="explanation" type="xs:string"/>
                <xs:element name="authorizedUsers" type="tAuthorizedUser"
minOccurs="0"/>
                <xs:element name="skipRule" type="QName" minOccurs="0"/>
            </xs:sequence>
        </xs:group>
        <xs:group name="ExtendedEdge">
            <xs:sequence>
                <xs:element name="optional" type="tOptionalMode"/>
                <xs:element name="explanation" type="xs:string"/>
```

```
                    <xs:element name="eventOrder" type="tEventOrder"/>
                    <xs:element name="timeEdge" type="xs:boolean"/>
                    <xs:element name="minTime" type="QName" minOccurs="0"/>
                    <xs:element name="maxTime" type="QName" minOccurs="0"/>
                    <xs:element name="escalationText" type="xs:string"/>
                </xs:sequence>
        </xs:group>
        <xs:group name="Jump">
                <xs:sequence>
                    <xs:element name="direction" type="tJumpDirection"/>
                    <xs:element name="sourceActivities" type="tJumpActivity"
maxOccurs="unbounded"/>
                    <xs:element name="targetActivities" type="tJumpActivity"
maxOccurs="unbounded"/>
                    <xs:element name="otherActivities" type="tJumpActivity"
minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="authorizedUsers" type="tAuthorizedUser"
minOccurs="0"/>
                </xs:sequence>
        </xs:group>
        <xs:group name="CriticalSection">
                <xs:sequence>
                    <xs:element name="concernedActivites" type="tExclusiveActivity"
minOccurs="0"/>
                </xs:sequence>
        </xs:group>
</xs:schema>
```

Hochschule Neu-Ulm
University of Applied Sciences

Eidesstattliche Erklärung

Ich versichere, dass ich die vorliegende Abschlussarbeit selbstständig angefertigt, nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe und die Überprüfung mittels Anti-Plagiatssoftware dulde.

Gundlfingen, 27.10.2022

Ort, Datum

Unterschrift