Institut für Parallele und Verteilte Systeme
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3064

Navigation in prozess-basierten kontextbezogenen Anwendungen unter Verwendung von unscharfen Zuständen

Jonas Palauro

Studiengang:	Softwaretechnik
Prüfer:	Prof. Dr. Kurt Rothermel
Betreuer:	Dipl.-Inf. Hannes Wolf
begonnen am:	1. Juni 2010
beendet am:	1. Dezember 2010
CR-Klassifikation:	H.4.1, I.2.3

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Geschäftsprozessmanagement hat sich zu einer der wichtigsten Informations-Technologien für unternehmen entwickelt. Die Verwendung von Prozessen beschränkt sich jedoch nicht mehr nur auf starre Vorgänge, die auf einer zuverlässigen Informationsbasis arbeiten können. Das Einsatzgebiet prozessbasierter Anwendungen erweitert sich auf Szenarien, in den der Anwender in seiner Tätigkeit unterstützt werden soll. Dem System stehen dabei keine direkten Nutzereingaben zur Verfügung, sondern es wird auf gesammelte Kontexinformationen zurückgegriffen.

Die Aktionen des Nutzers werden über verschiedene Sensoren erfasst und über ein Kontextsystem der Anwendung zur Verfügung gestellt. Die erfassten Daten müssen dafür zunächst aufbereitet und interpretiert werden, um festzustellen was für eine Aktion von dem Nutzer ausgeführt wurde.

Durch Umgebungseinflüsse oder eine ungenaue Datenerfassung wird die Interpretation der Kontextinformationen komplexer und es kann nicht mit absoluter Sicherheit festgestellt werden, was für eine Situation eingetreten ist. Ebenfalls können Situationen eintreten, in denen eine ausgeführte Nutzeraktion falsch oder gar nicht von dem Kontextsystem erfasst werden konnte.

Das führt dazu, dass die Datenerfassung nur ein unscharfes Ergebnis liefert, was in der weiteren Verarbeitung der Daten berücksichtig werden muss. Es kann der Fall eintreten, dass der Prozessnavigation nur unvollständige oder fehlerhafte Daten zur Verfügung stehen.

Die Ausführung eines Prozesses muss unter diesen Voraussetzungen flexibel auf die gelieferten Kontexinformationen reagieren und mit unscharfen Daten umgehen können. Es werden unter anderem Algorithmen benötigt, die in der Lage sind unter unsicheren Eingabedaten robust zu navigieren. Zudem muss ermöglicht werden mit teilweise falschen oder fehlenden Informationen zuverlässig innerhalb eines Prozesses zu navigieren.

Die Verwendung von unscharfen Informationen in Geschäftsprozessen wird nur vereinzelt umgesetzt [See05]. Größtenteils sind die Systeme für die schnelle Abarbeitung von Transaktionen ausgelegt, die auf konkrete Eingabewerte und klar definierte Entscheidungsregeln zurückgreifen.
Wieland *et al.* [WKL+09] beschreibt einen Ansatz, bei dem ein Workflow System auf unscharfen Kontextinformationen arbeitet. Hier werden spezielle Anforderungen gestellt, welche Zuverlässigkeit die erfassten Kontextinformationen aufweisen müssen, damit sie von dem

Workflow System verarbeitet werden. Ist die geforderte Zuverlässigkeit nicht gegeben werden die Kontextinfomationen herausgefiltert.

In der vorliegenden Arbeit wird ein System umgesetzt, das unscharfe Zustände innerhalb eines Flows zulässt und mit unscharfen oder unvollständigen Eingabedaten zu einer korrekten Ausführung des Flows führt. Dabei müssen zudem fehlerhafte oder fehlende Informationen toleriert werden, um eine weitere Ausführung des Flows zu ermöglichen.

Das System wird auf dem *FlowCon* System [WHR10] aufbauen und einige hier verfügbare Komponenten verwenden, die nicht Teil dieser Arbeit sind. Dazu gehören im Wesentlichen der *Flowgenerator*, das *Kontextsystem* und der *Simulator*.

Ein mögliches Einsatzgebiet des Systems ist die Unterstützung von Benutzer bei ihrer Tätigkeit, indem die Aktionen über ein Kontextsystem erfasst werden und der Flownavigation als Eingabedaten geliefert werden. Eine Herausforderung dabei ist die Behandlung von unscharf erfassten Kontextinformationen und die Problematik diese Daten richtig zu interpretieren. So können Fälle eintreten in denen eine Aktion ausgeführt wurde, aber nicht korrekt von dem System erfasst oder interpretiert werden konnte. Das kann dazu führen, dass ein Flow nicht abgeschlossen werden kann, selbst wenn alle benötigten Aktionen ausgeführt wurde.

In Kapitel 2 werden für diese Arbeit relevante Themen eingeführt. Dazu gehören Kontextbezogene Anwendungen und ihre grundlegenden Eigenschaften. Dabei werden die unterschiedlichen Ausprägungen dieser Systeme dargestellt. Darauf aufbauend werden Ansätze für die unscharfe Datenverarbitung dargestellt. Anschließend wird die Fuzzy Logik eingeführt und anhand eines Beispiels dargestellt, wie sie in der Verarbeitung unsicherer Daten eingesetzt werden kann. Darauf folgenden werden vorhandene Navigationsverfahren vorgestellt, die auf unscharfen Daten arbeiten.

In Kapitel 3 wird die Ausgangssituation für die Umsetzung untersucht und die Anforderungen an das umzusetzende System formuliert. Dabei wird zunächst eine Systemübersicht gegeben, die Anwendung dargestellt und anschließend die konkreten Anforderungen erhoben.

In Kapitel 4 wird das System mit seinen zugehörigen Komponenten entworfen. Dabei wird zunächst eine Übersicht über das System gegeben und dann die Funktionen und Komponenten entworfen.

In Kapitel 5 wird das entwickelte System einer Evaluation unterzogen. Dafür wird das Testmodell definiert, die zu erwartenden Problemfälle erläutert und die Konfigurationen für die Testläufe aufgeführt.
Anschließedn werden die Ergebnisse der Evaluation präsentiert und diskutiert.

In Kapitel 6 werden die Ergebnisse dieser Arbeit zusammenfassend dargestellt und ein Ausblick auf weitere Entwicklungen in diesem Themenbereich gegeben.

2 Grundlagen

Die Verwendung von Prozessen soll nicht mehr nur auf starre Vorgängen beschränkt sein, die auf einer zuverlässigen Informationsbasis arbeiten, sondern das Einsatzgebiet um Szenarien erweiteren, in denen der Nutzer in seinen täglichen Tätigkeiten unterstützt wird. Dem Prozess stehen dabei nur erfasste Kontextinformatione und keine direkten Nutzereingaben zur Verfügung. Die Prozesse müssen dabei flexibel an die gegebenen Kontextinformationen angepasst werden und auch mit unscharfen Informationen umgehen können, die durch die indirekte Interaktion mit dem Nutzer auftreten können.
Aus diesen Gründen wird einerseits ein Verfahren benötigt, dass die erfassten Kontextinformationen analysiert und für die weitere Verarbeitung aufbereitet und ein Navigationsalgorithmus der auf diesen Daten arbeiten kann. Die besondere Herausforderung dabei ist, dass es sich bei den Kontextinformationen nicht um zuverlässige Informationen handelt, sondern diese einer gewissen Unschärfe unterliegen, die bei der Datenverarbeitung und der Navigation berücksichtigt werden müssen.
Arbeitet der Navigationsalgorithmus der prozess-basierten Anwendung auf unscharfen Daten, wird eine Toleranz für Fehlinterpretationen in der Datenaufbereitung und -verarbeitung benötigt, um den Ablauf robust zu halten.

In Abschnitt 2.1 werden kontextbezogene Anwendungen eingeführt. Dabei werden verschiedene Anwendungsszenarien geschildert und Probleme bezüglich der Datenerfassung beleuchtet. In Abschnitt 2.2 werden verschiedene Ansätze dargestellt, wie unsichere oder unscharfe Daten innerhalb einer Anwendung aufbereitet und verarbeitet werden können. Dazu werden in Abschnitt 2.3 Grundlagen der Fuzzy Logik eingeführt und welche Möglichkeiten sich für die Verarbeitung unsicherer Daten daraus ergeben. In Abschnitt 2.4 werden Methoden zur Navigation und Entscheidungsfindung basierend auf unsicheren Daten aufgeführt und diskutiert.

2.1 Kontextbezogene Anwendungen

Kontextbezogene Anwendungen werden dadurch definiert, dass sie Kontextinformationen, wie die Position eines Nutzers oder Objekts und die Umgebung, nutzt und dadurch beeinflusst wird. In der Literatur gibt es unterschiedliche Definitionen von Kontext und kontextbezogenen Anwendungen [DA99].

Definition: Kontext Kontext sind die Informationen, die die Situation oder Aktivität einer Entität beschreiben. Eine Entität ist eine Person, ein Objekt oder eine Position, die für das Verhalten der Anwendung relevant sind. Die Entität wird dabei als Teil des Kontexts interpretiert.

Diese Definition orientiert sich an bestehenden Kontextdefinitionen [DA99, Roto8] und beschreibt, was in dieser Arbeit als Kontextinformationen bezeichnet wird.

Eine Kontextbezogene Anwendung nutzt Kontextinformationen und wird von ihnen beeinflusst. In der Literatur werden dafür unterschiedliche Definitionen gegeben. Schilit und Theimer [ST94] beschränken ihre Definition darauf, dass die Anwendung über den Kontext informiert wird und sich entsprechend an diesen Kontext anpassen kann.
Andere Definitionen sind allgemeiner gehalten, sie definieren kontextbezogene Anwendungen damit, dass sie die Möglichkeit besitzen, Kontextinformationen zu erfassen, zu interpretieren und darauf zu reagieren [HNBr97, Pas98]. Spziellere Definitionen, im Bezug auf die Anpassung des Systems an einen gegebenen Kontext, beschreiben die kontextbezogene Anwendungen als Systeme, die dynamisch ihre Verhalten an den Kontext des Benutzers und der Umgebung anpassen [BBC97, DMC+98]. Brown [Bro98] beschreibt kontextbezogenen Anwendungen als Anwendungen, die mittels Sensoren den aktuellen Kontext eines Benutzers erkennen und entsprechende Aktionen ausführen oder Informationen bereitstellen kann. Fickas *et al.* [FKS97] definiert als *Enviroment-Directed Computing* ein System, das Änderungen in der Umgebung erfasst und entsprechend vordefinierter Richtlinien darauf reagiert.
Die Definitionen von Brown und Fickas *et al.* beschreiben dabei was im Rahmen dieser Arbeit unter kontextbezogener Anwendung verstanden wird am besten.

Kontextbezogene Systeme können anhand ihrer Ausprägungen kategorisiert werden [DA99, Roto8, ST94].
Kontextbasierte Präsentation: Anhand der verfügbaren Kontextinformationen wird die Anwendung unterschiedlich dargestellt. Das kann verschiedene Ausgabeformate wie Audio oder Visualisierung betreffen. Ein einfaches Beispiel dafür ist ein GPS-Navigationsgerät, das entsprechend der aktuellen Geschwindigkeit den angezeigten Kartenausschnitt vergrößert und die Lautstärke der Sprachausgabe erhöht oder die Displaybeleuchtung anhand der Lichtverhältnisse anpasst.
Kontextbasierte Auswahl: Basierend auf den Kontextinformationen können unterschiedliche Dienste ausgewählt und angeboten werden. So können anhand einer erfassten Benutzerlokation verfügbare Endgeräte wie z.B. ein Drucker oder Faxgerät an das System angebunden werden, oder Informationen zu in der Umgebung verfügbaren Diensten wie Reisemöglichkeiten oder Reservierungoptionen für Restaurants angeboten werden.
Kontextbasierte Ausführung: Wird ein bestimmter Kontext eines Objekts oder einer Person erkannt, können definierte Aktionen ausgeführt werden. Das kann das Stummschalten eines Mobiltelefons sein, wenn über die Position erkannt wird, dass die Person ein Besprechungs-

zimmer betritt, oder das Aus- und Einschalten der Fahrzeugscheinwerfer beim Einfahren und Verlassen eines Tunnel, wobei auf die Lichtverhältnisse reagiert wird.

Das im Rahmen dieser Arbeit entwickelte System fällt in die Kategorie der kontextbasierten Ausführung. Durch die Anbindung unterschiedlicher Sensoren soll ermöglicht werden eine Situation oder die Aktion eines Nutzers zu erkennen und basierend auf diesen Kontextinformationen Entscheidungen innerhalb des Navigationsalgorithmus zu treffen.

Werden die Kontextinformationen über Sensoren erfasst sind die Herausforderungen die Erfassung, Aufbereitung und Interpretation der verfügbaren Daten. Weniger komplexen Daten, wie die Geschwindigkeit oder Position eines Objekts mit GPS-Gerät oder der Temperatur mit einem Thermometer können einfach und zuverlässig erfasst werden. Sollen allerdings Situationen oder die Aktion eines Benutzers erkannt werden, wobei Daten von unterschiedlichen Sensoren erfasst und interpretiert werden müssen, lässt sich nicht sicher beurteilen, ob eine angenommene Situation oder Aktion eingetreten ist. Das liegt zum einen daran, dass durch Umgebungseinflüsse Rauschen entsteht, wodurch die Sensordaten beeinflusst werden und die Interpretation von unterschiedlichen Sensordaten und das Schließen auf eine bestimmte Situation nicht trivial sind.

Die Kontextinformationen werden nicht in dem entwickelten System erfasst, sondern von einem angebundenen Kontextsystem. Es ist dabei transparent, welche Sensoren für die Erfassung eingesetzt werden und wie die erfassten Daten con dem Kontextsystem interpretiert werden. Von dem Kontextsystem werden Events geliefert, die die Kontextinformationen enthalten und die Sicherheit angeben, mit der davon ausgegangen werden kann, dass die gelieferte Information korrekt erfasst wurde.

2.2 Unscharfe Datenverarbeitung

Bei der Erfassung von Sensordaten kann nicht davon ausgegangen werden, dass durchgängig exakte Werte erfasst werden. Besonders bei dem Einsatz mehrerer unterschiedlicher Sensoren muss berücksichtig werden, dass die erfassten Daten in einem gewissen Rahmen unscharf sind [HI04]. Diese Unschärfe muss in der weiteren Verarbeitung berücksichtigt werden, um möglichen Fehlinterpretationen vorzubeugen.
Eine Möglichkeit diese Unschärfe in der Datenerfassung zu berücksichtigen ist, die erfassten Informationen mit einem Zuverlässigkeitswert zu versehen, der angibt, wie mit welcher Sicherheit die Situation erfasst wurde [SAT+99, STL01]. Falls eine eindeutige Interpretation der Daten nicht möglich ist, kann eine Menge von Alternativen angegeben werden, bei der jeder ein Wert zugeordnet ist, der die Sicherheit der jeweiligen Alternative darstellt [Pra].

Mit der Information darüber, mit welcher Sicherheit eine Information erfasst wurde kann die Anwendung Anforderungen an die erfassten Daten stellen [LSI+] die für eine weitere

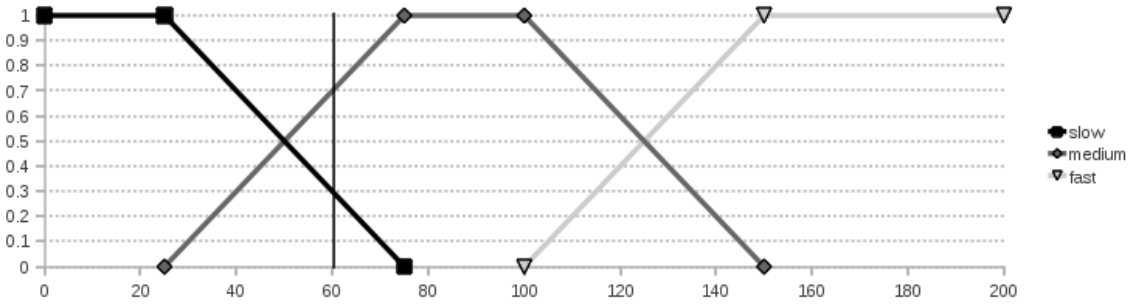

Abbildung 2.1: Zugehörigkeitsfunktion

Verarbeitung erfüllt werden müssen. Die entsprechende Reaktion des Systems nur erfolgt dann, wenn eine Situation mit der geforderten Sicherheit erfasst weren konnte.

2.3 Fuzzy Logik

Die Fuzzy Logik (engl. *fuzzy* unscharf) ist eine Theorie zur Modellierung von Unsicherheiten. Sie basiert auf den Fuzzy Mengen [Zad65] und Zugehörigkeitsfunktionen, mit den für ein Objekt oder einen Wert die Zugehörigkeit zu der Menge abgebildet werden kann.
Der entscheidende Punkt in der Fuzzy Logik ist, das bei dieser Abbildung nicht mehr nur auf *true* oder *false* abgebildet wird, ein Wert also der Menge angehört oder nicht. Der Wert wird mit der Zugehörigkeitsfunktion auf das Intervall $[0, 1]$ abgebildet, wobei 0 bedeutet, dass der Wert der Menge nicht angehoert und 1, dass er vollständig dieser Menge angehört. Des Weiteren ist es möglich, dass ein Wert zwei unterschiedlichen Mengen zugeordnet wird und dafür jeweils eine bestimmte Zugehörigkeit besitzt.

Wird die Geschwindigkeit eines Auto erfasst, so wird der Wert der gemessenen Geschwindigkeit dem System übermittelt. Soll die Geschwindigkeit von dem System als *langsam*, *mittel* oder *schnell* interpretiert werden, werden die Werte *a* und *b* definiert, ab welchem die Geschwindigkeitsinterpretation von *langsam* auf *mittel* und von *mittel* auf *schnell* wechselt. Für den Menschen ist diese Art von Interpretation nicht intuitiv, da in der natürlichen Sprache ein fließender Übergang zwischen den Bezeichnungen existiert und nicht über einen exakten Wert definiert ist. Mit Fuzzy Mengen können linguistische Variablen formuliert werden [Zad73], die sich an den Ausdrücken der natürlichen orientieren. Für das Beispiel des Geschwindigkeit kann die linguistische Variable *Geschwindigkeit* definiert werden, die die Werte *langsam*, *mittel* und *schnell* annehmen kann. Mit der Zugehörigkeitsfunktion aus Abbildung 2.1 wird die erfasste Geschwindigkeit auf die linguistische Variable *Geschwindigkeit* abgebildet. Einer gemessenen Geschwindigkeit von *60 km/h* wird mit einer Zugehörigkeit von 0.7 *mittel* und einer Zugehörigkeit von 0.3 *langsam* zugeordnet.

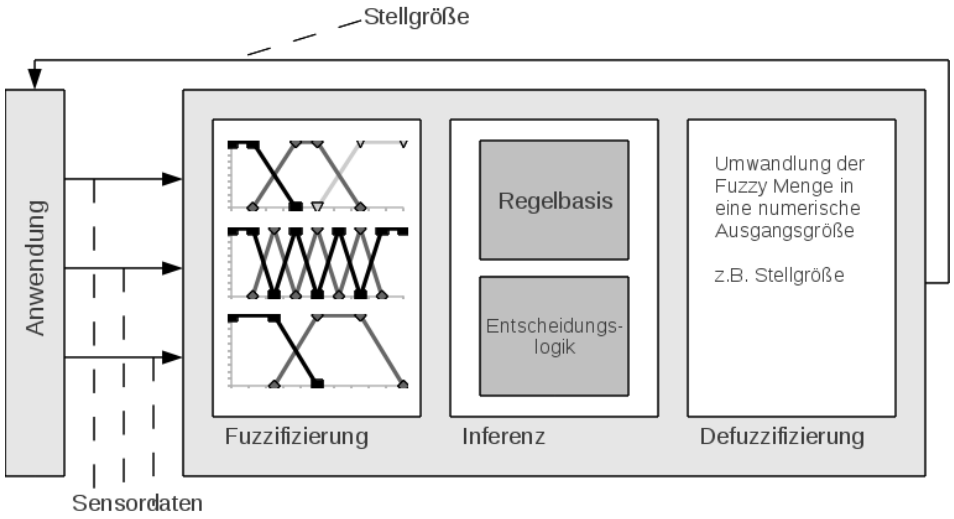

Abbildung 2.2: Komponenten eines Fuzzy Systems

Bei der Festlegung der Grenze auf einen festen Wert kann eine kleine Änderung des Eingabewerts eine Änderung der Interpretation bedeuten. Das wird durch die Vergabe der Zugehörigkeiten aufgeweicht und kann so leichte Schwankungen oder Ungenauigkeiten der Eingabedaten hinnehmen.

Im Wesentlichen besteht ein Fuzzy System aus drei Komponenten (siehe Abbildung 2.2). Der Fuzzifizierer bildet die Eingabedaten auf Fuzzy Mengen ab, das Inferenzsystem, das mehrere Eingabewerte annehmen kann und basierend auf deren Auswertung und den hinterlegten Regeln die Ergebniswerte erstellt und der Defuzzifizierer, der die Ergebniswerte des Inferenzsystems zu einem konkreten Ergebniswert zusammenstellt. Dieses Ergebnis kann dem System als Stellgröße zurückgegeben werden und kann somit Einfluss auf die weitere Ausführung nehmen.

Dem beschriebenen Beispiel der Geschwindigkeitserfassung wird ein weiterer Sensor hinzugefügt, der den Abstand zu dem vorausfahrenden Auto erfassen kann. Auch diese Daten werden mittels einer Zugehörigkeitsfunktion auf linguistische Variablen abgebildet. Der Abstand kann dabei die Werte *gering*, *mittel* und *groß* annehmen.
Damit soll das Bremsverhalten gesteuert werden, wobei als Ergebnis ebenfalls eine Fuzzy Menge ausgegeben wird, die eine Zugehörigkeitsfunktion zu den Werten *stark bremsen*, *leicht bremsen* und *nicht bremsen* liefert.
Ähnlich wie in der booleschen Logik können in der Fuzzy Logik Regeln definiert werden, die eine Schlussfolgerung ermöglichen. So können für dieses System die folgenden Regeln definiert werden:

1. *wenn* eigene Geschwindigkeit schnell *und* Abstand gering *dann* stark bremsen

2. *wenn* eigene Geschwindigkeit mittel *und* Abstand gering *dann* leicht bremsen
3. *wenn* eigene Geschwindigkeit gering *und* Abstand mittel *dann* nicht bremsen

Durch die erfassten Sensordaten und deren Interpretation wird bestimmt, welche Regeln angewendet werden.
Die Funktion $\mu_x()$ gibt die Werte der Zugehörigkeitsfunktion an, die für das Beispiel folgendermaßen definiert sind.

- $\mu_{gering}(60) = 0.3$; $\mu_{mittel}(60) = 0.7$; $\mu_{schnell}(60) = 0.0$
- $\mu_{gering}(20) = 0.75$; $\mu_{mittel}(20) = 0.25$; $\mu_{gro}(20) = 0.0$

Für die anzuwendenden Regeln gilt, dass bei *und*-Operatoren, das Minimum ($min(\mu_1, \mu_2)$) und bei *oder*-Operatoren das Maximum ($max(\mu_1, \mu_2)$) der Eingabeddaten geliefert wird. Damit ergibt sich folgende Ergebnismenge.

1. $\mu_{starkbremsen} = min\{\mu_{schnell}(60) = 0.0, \mu_{gering}(20) = 0.75\} = 0.0$
2. $\mu_{leichtbremsen} = min\{\mu_{mittel}(60) = 0.7, \mu_{gering}(20) = 0.75\} = 0.7$
3. $\mu_{nichtbremsen} = min\{\mu_{gering}(60) = 0.3, \mu_{mittel}(20) = 0.25\} = 0.25$

Die Fuzzy Menge die als Ergebnis geliefert wird muss abschließend mittels Defuzzifizierung auf einen konkreten Wert abgebildet werden, der für die weiter Verarbeitung von dem System verstanden werden kann. Die Einfachste Methode der Defuzzifizierung ist die Maximum-Defuzzifizierung, mit welcher das Ergebnis mit dem maximalen Zugehörigkeitswert gewählt wird. In dem beschriebenen Beispiel würde sich damit als Ergebnis *leicht bremsen* ergeben, da $max\{\mu_{starkbremsen} = 0.0, \mu_{leichtbremsen} = 0.7, \mu_{nichtbremsen} = 0.25\} = leichtbremsen$.

Innerhalb eines Fuzzy Systems sind keine harten Übergänge für die Abbildung der Eingabewerte gesteckt. Kleine Veränderungen der Eingabedaten haben nur eine entsprechend geringe Auswirkung auf die Auswertung. Für die Interpretation von Sensordaten, bietet das den Vorteil, dass unscharfe Daten und Umgebungseinflüsse besser toleriert werden können. Wie in dem Beispiel angedeutet, lassen sich auch Konbinationen von unterschiedlichen Datenquellen realisieren. Dabei wird für jede Datenquelle eine Fuzzifizierungsfunktion angegeben, es werden Regeln auf die Entscheidungslogik definiert und eine Funktion zur Defuzzifizierung angegeben.

2.4 Navigation basieren auf unscharfen Daten

Wie anhand des Beispiels in Abschnitt 2.3 dargestellt wurde, biete die Fuzzy Logik einen Ansatz wie basierend auf unscharfen Informationen und wenigen Regeln Entscheidungen getroffen werden können.

Im Folgenden werden bestehenden Ansätze dargestellt, die mit unsicheren Informationen navigieren oder Entscheidungen treffen.

Thomas und Adam [Ada, Loo06] beschreiben den Einsatz von Fuzzy Logik in der Modellierung von Geschäftsprozessen und welche Verbesserungen damit erreicht werden können. Oft existieren keine exakten Werte oder mathematische Modelle auf deren Basis klare Regeln definiert und Entscheidungen getroffen werden können. Verbale Aussagen oder unklar definierte Bedingungen können nicht auf eindeutige Werte abgebildet werden und Entscheidungen werden basierend auf unscharfen Daten und Abhängigkeiten zwischen verschiedenen Informationen getroffen.
Als Beispiel wird die Prüfung eines Kundenauftrags angeführt. Dabei werden die folgenden vier Bedingungen überprüft und basierend auf dem Resultat eine Entscheidung getroffen, ob der Auftrag angenommen oder abgelehnt wird.

- technische Machbarkeit
- wirtschaftliche Machbarkeit
- Kreditwürdigkeit des Kunden
- Verfügbarkeit des Produkts

Der Auftrag wird in diesem Beispiel nur dann angenommen, wenn alle Bedingungen erfüllt sind.
Ein Nachteil dieses Verfahrens ist, dass ein negatives Ergebnis in der Auswertung automatisch zu einer Ablehnung des Auftrags führt. Alle anderen Werte werden dabei nicht berücksichtigt und es ist nicht möglich durch einen guten Wert in einer Kategorie einen anderen auszugleichen. Die Beurteilung durch einen Menschen könnte ein Ungleichgewicht in den einzelnen Auswertungen anders interpretieren und den Auftrag vergeben.
Werden die Bedingungen nicht mehr nur mit *ja* oder *nein* beantwortet, sondern als Fuzzy Menge dargestellt, bieten sich umfangreichere Möglicheiten zu Entscheiden, ob der Auftrag angenommen oder abgelehn wird. Es ist möglich Abhängigkeiten zwischen den Einzelergebnissen zu definieren, die es erlauben, Werte gegeneinander Aufzuwiegen und über definierte Regeln das verfügbare Wissen in den Entscheidungsprozess zu integrieren.

In Workflow Systemen wird die Verwendung von unscharfen Daten bislang wenig berücksichtig. Die generelle Auslegung ist, viele Transaktionen in kürzester Zeit auszuführen und dabei auf exakte Werte und klar definierte Entscheidungregeln zurückgreifen zu können. Kommen unsichere Daten zum Einsatz, müssen diese vor der weiteren Verarbeitung auf exakte Werte oder mathematische Formeln abgebildet werden. Bei dieser Übersetzung entsteht das Problem, dass relevante Informationen verloren gehen können und dadurch die Effizienz negativ beeinflusst werden kann [Ada03, See05].

Wieland *et al.* [WKL+09, WKNL07] beschreibt einen Ansatz zur Integration und Verwendung unscharfer Sensordaten in Workflows und wie die Problematik der unsicheren Daten behandelt werden kann. Auf der einen Seite werden Sensordaten erfasst und mit technischen

Metadaten annotiert und auf der anderen Seite stellt das Workflow System hohe Anforderungen an die Zuverlässigkeit der gelieferten Informationen. Diese Lücke muss durch ein Kontextsystem geschlossen werden, das die Verarbeitung von *Quality of Context (QoC)* [BS03, KH05] unterstützt.
Bei der Datenerfassung werden in die Metadaten Informationen über die Qualität der erfassten Daten geschrieben. Diese beinhalten Informationen über die Genauigkeit der erfassten Daten, aufgetretene Fehler in der Datenverarbeitung, Zuverlässigkeit des Sensors, Fehler die durch Umgebungseinflüsse aufgetreten sind und die Auflösung die der Sensor für die Datenerfassung verwendet.
Auf der anderen Seite werden von dem Workflow System Anforderungen an die gelieferten Kontextinformationen gestellt. Werden diese nicht erfüllt, werden die entsprechenden Daten herausgefiltert und nicht weiter verarbeitet. Diese Anforderungen umfassen Angaben über die Aktualität der Daten, die Genauigkeit der Werte, die Qualität der Digitalisierung des erfassten Signals, die Wahrscheinlichkeit, dass die gemessenen Daten korrekt sind und die Zuverlässigkeit des entsprechenden Sensors.
Für eine erfasste Kontextinformation muss geprüft werden, ob die eingetragenen Metadaten die von dem Workflow System gestellten Anforderungen erfüllt. Dafür wird ein *matchmaking* durchgeführt, das die Metadaten gegen die Anforderungen prüft. Nur wenn diese erfüllt sind werden die Daten ausgeliefert und verarbeitet.

3 Analyse und Anforderungen

Nach der Einführung in die für diese Arbeit wichtigen Grundbegriffe und Verfahren der Fuzzylogik, der Verarbeitung und Aufbereitung von unscharfen Daten und der Verfahren bei der Navigation wird nun in Abschnitt 3.2 ein System vorgestellt in das der Navigationsalgorithmus integriert werden soll. Dieses System beinhaltet die Generierung eines Flowmodells und ein Kontextsystem, das den Navigator mit den für die Durchführung benötigten Kontextinformationen in Form von Events versorgt. Basierend darauf werden die Probleme analysiert, die durch Unschärfe in der Datenversorgung entstehen können und in Abschnitt 3.5 die Anforderungen an den Navigationsalgorithmus gezeigt. Die Anforderungen werden dabei für die Analyse und Aufbereitung der Datensätze und die Verarbeitung und Navigation getrennt.

3.1 Beispielszenario

Das hier betrachtete Beispielszenario beschreibt die Navigation innerhalb eines Flowmodells, bei dem die Durchführung einzelner Benutzeraktivitäten durch ein Kontextsystem erfasst und in Form von Events dem System übergeben werden. Das Flowmodell definiert die die Aktivitäte und Transitionen sowie die erwarteten Events der Aktionen, die für den Abschluss einer Aktivität benötigt werden. Die Aktivitäten können eine oder mehrere Events für den Abschluss benötigen, als optional gekennzeichnet sein oder als Entscheidungsknoten innerhalb des Flows fungieren und basierend auf den empfangenen Eventdaten den Pfad definieren, der bei der Ausführung verfolgt werden muss.

Während der Ausführung werden die, von dem Benutzer ausgeführten Aktivitäten von dem Kontextsystem erfasst und dem Navigationsalgorithmus ausgeliefert. Die Aufgabe hier besteht darin, die Informationen auszuwerten, der korrekten Aktivität des Flows zuzuordnen und entsprechend innerhalb des Flows weiter zu navigieren.

3.2 Systemübersicht

Im Folgenden wird ein System beschrieben, das die Auführung eines in Abschnitt 3.1 beschriebenen Szenarios ermöglicht. Zunächst werden die zentralen Komponenten (siehe

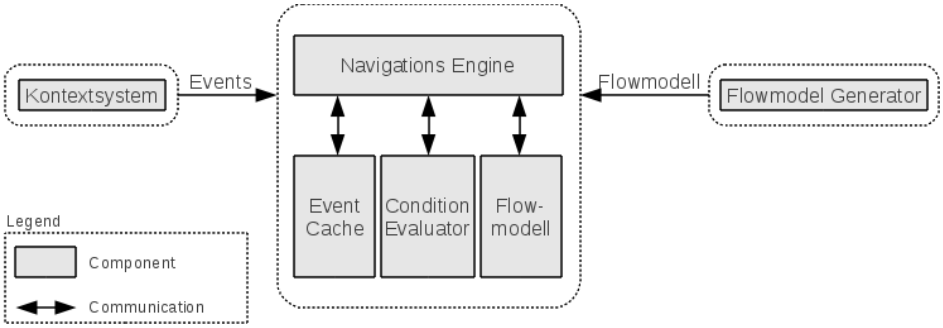

Abbildung 3.1: Systemübersicht

Abbildung 3.1) des Systems und deren Aufgaben und Funktionen beschrieben und anschließend diskutiert, in welchen Situationen Probleme entstehen können wenn man von der Annahme ausgeht, dass Benutzeraktionen, die von dem Kontextsystem erfasst werden unscharf oder fehlerhaft sein können.

3.2.1 Kontextsystem

Das Kontextsystem dient zur Erfassung und Bereitstellung von Kontextinformationen. Das wird dadurch motiviert, das die Ausführung des Flows durch die Wahrnehmung von Benutzeraktivitäten gesteuert werden soll.
Für die Erfassung der Benutzeraktivitäten können unterschiedliche Sensoren eingesetzt werden. Zu berücksichtigen ist, dass die Erfassung mittels Sensoren immer mit einem gewissen Maß an Unschärfe behaftet ist, das kann durch geringe Abtastraten oder Umgebungseinflüsse entstehen. Folglich kann bei der weiteren Verarbeitung der Kontextinformationen nicht von exakten Eingabedaten ausgegangen werden und die Unschärfe der Erfassung muss weiter berücksichtig werden.

Die erfassten Sensordaten werden in dem Kontextsystem aufbereitet, in Events verpackt und dem Navigationsalgorithmus ausgeliefert.

3.2.2 Flowmodell

Das Flowmodell ist die Vorlage der Auszuführenden Schritte der Anwendung (siehe Abbildung 4.2), dabei werden die Aktivitäten und ihre Abhängigkeiten definiert. Das beinhaltet die Definition der einzelnen Aktivitäten und die Spezifikation ob und welche Events von dem Kontextsystem für den Abschluss der Aktivität erforderlich sind, die Transitionen zwischen den Aktivitäten und Bedingungen, die zum Aktivieren bestimmter Transitionen erforderlich sind.

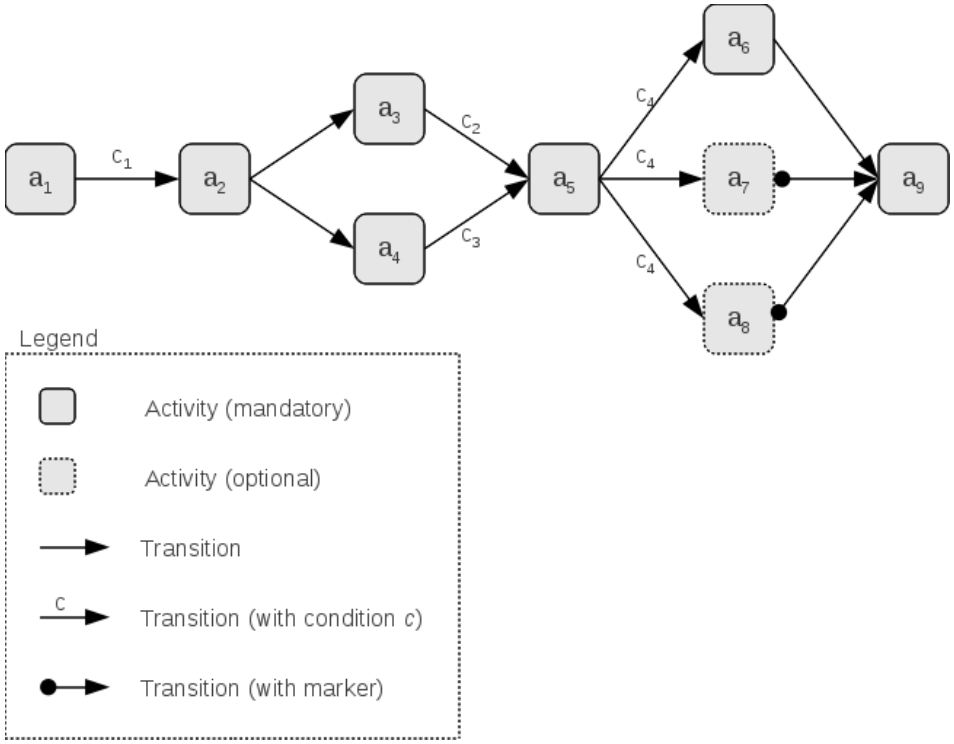

Abbildung 3.2: Flowmodell

Das Flowmodell wird als gerichteter azyklischer Graph definiert mit den Aktivitäten als Konten und den Transitionen als Kanten, wobei die Bedingungen dabei direkt den Transitionen zugeordnet werden.

Eine Aktivität definiert einen auszuführenden Arbeitsschritt, dieser kann eine externe Benutzeraktion erwarten oder die Ausführung einer Berechnung oder eines Services.
Erwartet eine Aktivität ein oder mehrere Benutzeraktionen, werden die entsprechenden Eventtypen definiert, die an die Aktivität ausgeliefert werden müssen.
Die Aktivität kann als *mandatory* gekennzeichnet werden, dadurch werden die Pflichtpfade in dem Flow definiert und es kann die Navigation (siehe subsection 3.2.3) beeinflusst werden.

Eine Transition $t = (a_x, a_y)$ definiert eine gerichtete Kante zwischen zwei Aktivitäten a_x und a_y. Einer Transition kann eine Bedingung zurgeordnet werden, die für die Aktivierung der Transition erfüllt werden muss. Bei einer Aktivität mit mehreren ausgehenden Transitionen kann durch die Auswertung der Bedingungen entschieden werden, welche Transition aktiviert wird - also welche Aktivität als nächstes ausgeführt wird. Die Auswertungen der Bedingungen basiert auf den empfangenen Events und Ergebnissen der Aktivität und wird in subsection 3.2.3 beschrieben.

Zudem können Transitionen mit einem *Marker* versehen werden der definiert, dass die Transition nicht aktiviert werden muss. Werden bei einer Aktivität mehrere Zweige des Ablaufs zusammengeführt, bei der nicht alle ausgeführt sein müssen, so können die eingehenden Transitionen der optionalen Zweige mit einem *Marker* gekennzeichnet werden. Dadurch wird für die Aktivität definiert, welche eingehenden Transitionen explizit aktiviert sein müssen, damit diese in den aktivien Zustand wechseln darf.

3.2.3 Navigationsalgorithmus

Der Navigationsalgorithmus ist die zentrale Komponente zur Steuerung des Ablaufs während der Ausführung, hier werden die Zustände des Flows verwaltet und überwacht sowie Kontextinformationen von dem Kontextsystem empfangen und deren Verteilung innerhalb des Flowmodells angestoßen.

Die Navigation innerhalb des Flows basiert auf der Auswertung der aktuellen Zuständen der Aktivitäten und Transitionen des Flowmodells. Für die Ermittlung der jeweiligen Zustände wird analysiert ob alle Teile der Aktivität ausgeführt wurden oder alle benötigten Benutzeraktivitäten erfasst und empfangen wurden oder die Bedingungen der Tansition erfüllt sind.

Allgemein wird während der Ausführung zwischen drei abstrakten Zuständen jeder einzelnen Aktivität unterschieden.

Inactive - Wird ein neues Flowmodell dem Navigationsalgorithmus übergeben, befinden sich zunächst alle Aktivitäten im Zustand *Inactiv*. Die Aktivitäten bleiben solange in diesem Zustand, bis sie von dem Navigationsalgorithmus, basierend auf dem aktuellen Fortschritts der Bearbeitung aktiviert werden.
Inaktive Aktivitäten werden nicht über empfangene Events informiert und können keine Events zugeordnet bekommen.

Active - Aktivitäten in dem Zustand *Active* befinden sich aktuell in Bearbeitung. Das bedeutet, dass sie über empfangene Events informiert werden, diese zugeordnet bekommen können, sofern sie die Bedingungen erfüllen und nach dem Empfang aller benötigten Events abgeschlossen werden können.

Complete - Wenn die Bearbeitung einer Aktivität abgeschlossen ist und sie alle benötigten Events empfangen hat wird der Zustand auf *Complete* gesetzt. Im Folgenden wird für die ausgehenden Transitionen geprüft, ob für das aktivieren der Transition die Erfüllung einer Bedingung gefordert ist. Ist das nicht der Fall oder die Bedingungen sind erfüllt, kann die Transition geschaltet werden und die folgende Aktivität vom Zustand *Inactive* auf *Active* gesetzt werden.

Nach der Übergabe des Flowmodells an den Navigationsalgorithmus werden die Startaktivitäten ermittelt und aktiviert, als Startaktivitäten werden die Aktivitäten ermittelt, die keine eingehenden Kanten besitzen.

Während der Ausführung, wird der Navigationsalgorithmus von dem Kontextsystem mit Eventdaten versorgt. Die zum Zeitpunkt des Eintreffens eines Events aktiven Aktivitäten werden darüber informiert und es wird geprüft, welcher Aktivität das entsprechende Event zugeordnet werden muss.
Für die Aktivitäten wird nach dem Empfang eines neuen Events geprüft, ob alle benötigten Events empfangen wurden und die Aktivität abgeschlossen werden kann.
Nach Abschluss einer Aktivität wird für die ausgehenden Transitionen geprüft ob für spezielle Bedinungen vorgegeben sind. Ist dies nicht der Fall, kann die Transition geschaltet und die folgende Aktivität aktiviert werden. Sind für eine oder mehrere ausgehenden Transitionen Bedingungen vorgegeben die erfüllt sein müssen werden diese gesondert ausgewertet und ensprechend geschaltet.
Bei einer Aktivität mit mehreren ausgehenden Transitionen können diese Bedingungen dazu verwendet werden Entscheidungskriterien zu definieren welchem Pfad für die weitere Ausführung gefolgt werden soll.

3.3 Anwendung

Im Folgenden wird eine Übersicht gegben, wie ein Flow ausführt wird bei dem Aktivitäten von Benutzeraktionen abhängig sind, die von dem Kontextsystem erfasst und dem System übergeben werden.

Zunächst wird das Flowmodell erstellt und die initialisiert, wobei die Startaktivitäten ermittelt und aktiviert werden. Während der Ausführung werden die Benutzeraktivtäten von den Sensoren erfasst und von dem Kontextsystem aufbereitet und dem Navigationsalgorithmus übergeben. Dieser interpretiert die empfangenen Eventdaten und ordnet sie den entsprechenden Aktivitäten zu. Nach dem Empfang neuer Eventdaten wird geprüft, ob die Aktivität alle benötigten Events erhalten hat. Ist dies der Fall werden die Bedingungen der ausgehenden Transitionen ausgewertet und die aktiviert deren Bedingungen erfüllt sind. Dadurch wird der nächste auszuführende Schritt innherhalb des Flows ermittelt und die entsprechenden Aktivitäten aktiviert und für den Empfang weiterer Events des Kontextsystems freigeschaltet.

Durch die Gegebenheit, dass die Erfassung der Benutzeraktivitätetn ein gewisses Maß an Unschärfe aufweist, dass in der Verarbeitung berücksichtig werden muss ergeben sich folgende konkrete Problemfälle die bei der Ausführung berücksichtigt und gelöst werden müssen.

- *Fehlende Events* - Benutzeraktivitäten die von dem Kontextsystem nicht erfasst wurden oder nicht die benötigte Genauigkeit aufweist, dass das System erkennen kann, dass eine bestimmte Aktion von dem Benutzer ausgeführt wurde. Das kann dazu führen, dass die Aktivitäten innerhalb des Flows, die auf das entsprechende Event warten nicht abgeschlossen werden können und so eine weitere Ausführung des Flows nicht möglich ist. Dadurch wird die Ausführung fälschlicherweise unterbrochen und der Flow kann nicht abgeschlossen werden, auch wenn alle Aktionen von dem Benutzer korrekt ausgeführt wurden.
- *Falsche Events* - Durch Rauschen oder Umgebungseinflüsse kann es passieren, dass die Sensoren Daten erfassen, die von dem Kontextsystem fälschlicherweise als Benutzeraktion interpretiert werden und dem Navigationsalgorithmus als erfasstes Event übergeben werden.
- *Fehlerhafte Auswertungen* - Wie in Abschnitt 3.2.1 beschrieben, wird anhand der erfassten Daten ausgewertet, welche Aktion die höchste Wahrscheinlichkeit hat. Das kann dazu führen, dass bei nicht eindeutigen Datensätzen in Kombination mit Rauschen oder Umgebunseinflüssen eine falsche Annahme getroffen wird und so die Ausführung einer anderen Benutzeraktion angenommen wird, als eingetreten ist. Anhand dieses einzelnen Datensatzes ist dieses Problem nicht zu erkennen, wird z. B. im ersten Schritt die Aktion a_2 erkannt, aber es ist Aktion a_1 ausgeführt worde, lässt sich das anhand der Daten nicht identifizieren. Wenn im nächsten Schritt die Aktion a_2 Ausgeführt wird und dabei korrekt die Aktion a_2 erkannt wird entsteht die Situation, dass das System zwei Events für die Aktion a_2 empfangen hat, allerdings keines für a_1.
  Basierend auf beiden empfangenen Eventdatensätzen muss hier entschieden werden können, bei welchem Event es sich um Aktion a_1 bzw. um a_2 handeln könnte.

Falsche Events, also zusätzliche oder doppelt auftretende Events sind durch einen optimistischen Ansatz relativ einfach zu tolerieren. Wenn man von der Annahme ausgeht, dass alle Benutzeraktionen korrekt ausgeführt werden und das Hauptproblem in der Erfassung besteht kann ein doppelt empfangenes Event ignoriert werden um eine weitere Ausführung des Flows zu gewährleisten.
In Kombination mit der Gegebenheit, dass Aktionen falsch interpretiert werden können, muss allerdings beachtet werden, dass ein als doppelt interpretierte Aktion eventuell nicht korrekt ausgewertet wurde, sondern eine andere Aktion ausgeführt wurde.

Ein *fehlendes Event* kann dazu führen, dass die Ausführung unterbrochen und nicht fortgesetzt werden kann. Wenn eine Aktivität nicht alle benötigten Events erhält, kann sie nicht abgeschlossen werden und so die folgenden Aktivitäten aktivieren. Mit der Annahme, dass das Problem in der Erfassung der Benutzeraktionen liegt, kann das dadurch entstehen, dass bei der entsprechenden Ausführung die Aktion nicht mit der geforderten Sicherheit erfasst wurde und so als nicht druchgeführt angenommen wird.

Werden Daten fehlerhaft Interpretiert, lässt sich das anhand des einzelnen Datensatzes nicht feststellen und muss basierend auf dem aktuellen Zustand der Ausführung analysiert und ausgewertet werden. Zusätzlich kann es dabei dazu kommen, dass Aktivitäten fälschlicherweise abgeschlossen wurden, was zu einem späteren Zeitpunkt rückgängig gemacht werden muss.

3.4 Allgemeine Anforderungen

Basierend auf den möglichen Problemen werden im Folgenden allgemeine Anforderungen gestellt, die erfüllt werden müssen um diese Probleme währende der Ausführung lösen zu können. Dabei werden Anforderungen für die Analyse und Aufbereitung der erfassten Daten und für die Entscheidungsfindung innerhalb des Navigationsalgorithmus erhoben.

Bei der Erfassung der Kontextinformationen muss berücksichtig werden, dass die Sensordaten einer Unschärfe unterliegen. Mit den verfügbaren Informationen muss nun das wahrscheinlichste Ergebnis ermittelt und für die weitere Verarbeitung aufbereitet werden. Wie beschrieben, wird bei der Interpretation der Informationen analysiert, welche Situation mit der höchsten Wahrscheinlichkeit erfasst wurde.
In Abschnitt 3.3 wurde gezeigt, welche Probleme bei der Verarbeitung von unscharfen Daten entstehen können. Ausgehend davon werden die Anforderungen erhoben, die bei der Aufbereitung der Daten erfüllt werden müssen, um eine eine Torelanz gegenüber fehlerhaften Auswertungen zu ermöglichen.

Die Anforderungen für die Datenverarbeitung haben direkte Einfluss auf die an den Navigationsalgorithmus. Durch die Anforderung, dass die erfassten Datensätze zu einem späteren Zeitpunkt mit umfangreicheren Daten neu evaluiert und somit die Ergebnisse geändert werden können muss auch der Navigationsalgorithmus auf diese Änderungen reagieren können.
Wird eine Situation angenommen, das entsprechende Event innerhalb der Navigation verarbeitet und die Aktivität abgeschlossen. So werden die folgenden Aktivitäten aktiviert. Wir zu einem späteren Zeitpunkt festgestellt, dass die getroffenen Annahmen fehlerhaft war und diese Situation angepasst werden muss kann das Probelem entstehen, dass die Entscheidungen, die aufgrund einer fehlerhaften Annahme getroffen wurden revidiert werden müssen. Anschließen kann eine erneute Analyse erfolgen.
Es ist also notwendig innerhalb der Navigation einen Mechanismuss zu schaffen, der getroffene Entscheidungen und Zustandsänderungen in der Flowausführung bis zu einem bestimmten Punkt zurücknehmen kann und basierend auf einem stabilen Zustand die Zuordnungen der empfangenen Events erneut vornehmen und auf einem aktualisierten Informationstand treffen kann.
Eine weitere Anforderung ist die Toleranz gegenüber nicht erkannter Benutzeraktivitäten. Das kann durch Rauschen oder Umgebungseinflüsse bei der Erfassung entstehen. Wird eine

Aktivität nicht erkannt kann es bei der Ausführung der Flows dazu kommen, dass eine Aktivität, die auf das entsprechenden Event wartet nicht abgeschlossen werden kann und so eine weitere Ausführung nicht möglich ist. Es ist also notwendig einen Mechanismuss zu schaffen, der diese Eigenschaft berücksichtigt, solche Fehlerfälle toleriert und eine weitere Ausführung des Flows ermöglicht.

Als Ziel wird mit diesen Anforderungen formuliert, einen Navigationsalgorithmus zu entwerfen, der auf unscharfen Informationen innerhalb eines Flows navigieren kann und dabei die Eigenschaften der unsicheren Datenversorgung berücksichtigt. Im Speziellen bedeutet das, dass die gelieferten Kontextinformationen zunächst aufbereitet und interpretiert werden müssen und dem Navigationsalgorithmus für die verarbeitung übergeben werden. Dabei muss beachtet werden, dass die Daten solange verfügbar gehalten werden, bis davon ausgegangen werden kann, dass die getroffenen Entscheidungen nicht mehr geändert werden müssen. Der Navigationsalgorithmus muss die Verareitung der unschafen Daten weiterführen indem die Möglichkeit geschaffen wird, Entscheidungen zu revidieren und neu zu evaluieren, sollte sich in der Auswertung der Datensätze aufgrund zusätzlicher Informationen etwas geändert haben.

3.5 Anforderungen

In Abschnitt 3.3 und Abschnitt 3.4 wurden Problemszenarien und allgemeine Anforderungen an das zu entwerfende System formuliert. Im Folgenden werden diese Anforderungen in die Themenbereiche Analyse und Aufbereitung der Kontextinformationen, Verarbeitung und Verteilung der Daten, Entscheidungprozesse und Vorkehrungen zur Steigerung der Robustheit und Toleranz gegenüber unschafer oder fehlerhafter Daten gegliedert.

3.5.1 Datenanalyse- und aufbereitung

Die von dem Kontextsystem gelieferten Datensätze müssen nach einem Schema aufbereitet werden die die unscharfen Informationen in ein Format überführen, dass von dem Navigationsalgorithmus als Entscheidungsgrundlage interpretiert werden kann und die Möglichkeit bietet mehrere Datensätze zu vergleichen und basierend auf umfangreicheren Informationen zu entscheiden, welche Interpretation für einen Datensatz unter bestimmten Voraussetzungen am wahrscheinlichsten ist.

3.5.2 Navigationsalgorithmus

Der Navigationsalgorithmus muss die aufbereiteten Daten verarbeiten und interpretieren können und basierend auf diesen Informationen Navigationsentscheidungen treffen können.

Zudem werden für die in Abschnitt 3.3 identifizierten Probleme bei der Verwendung unscharfer Daten Vorkehrungen benötigt, die diese Fälle bei der Navigation berücksichtigen und möglichst tolerieren können.

Als konsequente Weiterführung der Anforderung, dass die Datensätze des Kontextsystems zu einem späteren Zeitpunkt auf einem umfangreicheren Informationsstand neu interpretiert und evaluiert werden können müssen diese möglichen Änderungen des Datenbestandes für die Navigationsentscheidungen berücksichtig werden. Im Falle einer Änderungen an Daten, die zu einer Entscheidung in der Navigation geführt haben, müssen die Entscheidungen revidiert und basierend auf einem aktualisierten Datenbestand neu evaluiert werden.
Für den Navigationsalgorithmus wird ein Mechanismus benötigt, der einen bestimmten Bereich des auszuführenden Flows identifiziert, in dem Änderungen auftreten können und dementsprechend Entscheidungen neu evaluiert werden müssen.

Es können Situationen eintreten, in denen der Navigationsalgorithmus mit fehlerhaften Daten von dem Kontextsystem versorgt wird. Das können Events sein, die nicht die benötigte Sicherheit aufweisen oder Events die fälschlicherweise erstellt wurden, obwohl keine Situation erfasst wurde. Ebenso können Fälle Auftreten in den eine Aktion von dem Kontextsystem nicht erfasst werden konnte und somit nicht an den Navigationsalgorithmus ausgeliefert wird.
Für diese Problemfälle muss die Navigation Mechanismen bereitstellen, die solche Fehlerfälle tolerieren können.

4 Entwurf

In Kapitel 3 wurden die Anforderungen an das System aufgeführt und welche Ziele erreicht werden sollen. Für die Umsetzung wird in diesem Kapitel eine Architektur beschrieben, die Funktionalität, die Komponenten und Abläufe spezifiziert sowie die Einschränkungen für die Umsetzung diskutiert.

Das System soll die Verarbeitung von unscharfen Kontextinformationen anbieten indem diese Kontextinformationen mit Methoden aus der Fuzzy Logik analysiert und aufbereitet werden. Die Navigation soll auf den so aufbereiteten Datensätzen arbeiten und dabei berücksichtigen, dass sich die Auswertung der ursprünglichen Daten basierend auf einem umfangreicherren Informationsbestand ändern kann. Das bedeutet, dass die Navigation während der Ausführung so flexibel gestaltet werden muss, dass eine Änderung der Navigationsentscheidungen basierend auf einem aktualisierten Datenbestand ermöglicht werden muss. Zudem muss eine Toleranz gegenüber der unscharfen Eingabedaten geschaffen werden, die Raum für Fehlinterpretationen oder fehlende Datensätze zulässt und es ermöglicht unter diesen Umständen einen Flow korrekt durchzuführen.

Die folgenden Abschnitte gliedern sich wie folgt:

4.1 Systemübersicht

In Abschnitt 3.2 wurde ein Systems beschrieben, in das die Funktionalität integriert werden soll. Im Folgenden wird diese Architektur aufgegriffen und die Komponente und Abläufe entworfen, die für die Umsetzung der gestellten Ziele erforderlich sind.

4.1.1 Zustände der Aktivitäten

In subsection 3.2.3 wurden bereits abstrakt die Zuständen beschrieben, die die Aktivitäten während der Auführung annehmen können und auf den die Navigation aufgebaut werden soll. Unter Berücksichtigung darauf, dass unscharfe Daten verarbeitet werden und die Interpretation eines einzelnen Datensatzes zu einem späteren Zeitpunkt geändert werden kann und das vereinzelt nicht oder nicht korrekt erfasste Daten von der Navigation toleriert werden sollen sind diese Zustände nicht mehr ausreichend und es lassen sich die Anforderungen nicht erfüllen.

Die bereits beschriebenen Zustände *Inactive*, *Active* und *Complete* bleiben weiterhin bestehen und auch an der Semantik der Zustände wird nichts verändert. Die Notwendigkeit zusätzlicher Zustände ergibt sich daraus, dass die Zustände weiterhin eindeutig definiert sein müssen. Wenn eine Aktivität den Zustand *Complete* annimmt dürfen keine weiteren Änderungen an dieser Aktivität vorgenommen werden und wird nicht weiterhin mit neu eintreffenden Events versorgt.

Zwischen den Zustandsübergängen von *Inactive* auf *Active* und *Active* auf *Comlete* werden jeweils ein weiterer Zustand eingefügt.
Der Zustand *Can Complete* wird zwischen *Active* und *Complete* eingefügt und ist folgendermaßen definiert. Eine Aktivität, die für jedes erwartete Event mindestens eines Empfangen hat, das die notwendigen Voraussetzungen erfüllt und noch nicht an eine andere Aktivität vergeben ist wird in den Zustand *Can Complete* versetzt. Im Gegensatz zum Zustand *Complete* wird die Aktivität weiterhin mit eigehenden Events versorgt.
Der Zustand *Prepare* wird zwischen *Inactive* und *Active* eignefügt. Eine Aktivität wird in diesen Zustand versetzt, wenn die Vorgängeraktivität auf den Zustand *Active* gesetzt wird. Bei dieser Zustandsänderung die die Aktivität bereits für den Empfang von Events registriert. Der Zweck dieses Zustands ist einen Vorausblick innerhalb des Flowmodells zu erhalten. Dadurch, dass die Aktivität bereits über Events informiert wird, kann hier schon evaluiert werden, ob sie theoretisch abgeschlossen werden könnte. Das hat den folgenden Hintergrund. Sollte die Vorgängeraktivität ein benötigtes Event nicht erhalten und kann somit nicht abgeschlossen werden, das toleriert werden und die Aktivität wechselt von *Prepare* direkt in den Zustand *Can Complete*.

4.1.2 Zustand des Flowmodells

Zeigt den aktuellen Zustands des Gesamtflows. Grundlegend können die Aktivitäten anhand ihres Zustands gruppiert werden und geben damit ebenfalls Information über den Fortschritt und den Bereich der Unsicherheit wieder.
Die Aktivitäten im Zustand *Complete* wurde korrekt und vollständig abgearbeitet, hier können keine Änderungen mehr ausgeführt werden.
Die Aktivitäten in den Zuständen *Can Complete*, *Active* und *Prepare* sind aktuell in Bearbeitung und für den Empfang von Eventdaten registiert. Die Aktivitäten in diesen Zuständen bilden den aktiven Bereich des Flowmodells innerhalb welchem die Zuordnung von Events und die Interpretation der Eventdaten geändert werden kann und Entscheidungen der Navigation revidiert und neu evaluiert werden können. Sie bilden während der Ausführung den unsicheren Bereich des Flows, der basierend auf den aktuell zur Verfügung stehenden Informationen den Zustand wiederspiegelt, sich aktuell noch in Bearbeitung befinden und im weiteren Verlauf geändert werden kann.

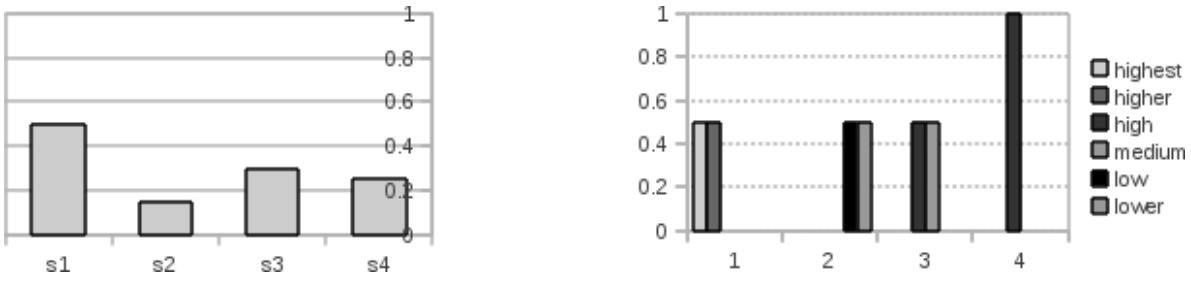

Abbildung 4.1: Datensätze

4.1.3 Datenhaltung

Die von dem Kontextsystem gelieferten Informationen werden zunächst aufbereitet und solange vorgehalten, bis für den entsprechenden Datensatz eine sichere Zuordnung vorgenommen werden konnte. Unter der Voraussetzungen, dass sich die Interpretation eines gelieferten Datensatzes und somit die Zuordnung des Events zu einer Aktivität während der Ausführung ändern kann, müssen die Daten solange verfügbar sein, bis keine Änderung mehr vorgenommen werden muss. Das wird dadurch definiert, dass die Aktivität, der das Event zugeordet wurde abgeschlossen werden kann und in den Zustand *Complete* versetzt wird.

4.2 Funktionalität

4.2.1 Datenverarbeitung

In Abschnitt 3.5.1 wurden grundlegende Anforderungen an die Aufbereitung der Kontextinformationen formuliert.
Der gelieferte Datensatz des Kontextsystems wird mit Hilfe eines Fuzzifizieres (vgl. Abschnitt 4.3.1) analysiert und entprechend definierter Regeln abgebildet. Dabei werden für das Eintreten einer bestimmten Aktion Fuzzy-Werte vergeben wie sicher von der entsprechenden Situation ausgegangen werden kann.

Die von dem Kontextsystem gelieferten Informationen, die für möglich eingetretene Nutzeraktionen einen Wahrscheinlichkeitverteilung enthalten (siehe Abbilung 4.1 links) werden in eine Fuzzy Menge überführt. Dabei wird für für die möglichen Aktionen die Zugehörigkeit einer liguistischen Sicherheitsvariablen vergeben (siehe Abbildung 4.1 rechts). Nach der Fuzzifizierung der Kontextinformationen können auf diesen Daten Regeln der Fuzzy Logik (Abschnitt 2.3) angewendet werden.

Wichtig für die Datenverarbeitung ist, dass möglichst wenig der Ursprungsdaten verloren gehen, falls diese zu einem späteren Zeitpunkt für eine erweiterte Analyse benötigt werden. Das kann der Fall sein, wenn nach einer ersten Analyse für zwei Datensätze das selbe Resultat angenommen wird und entschieden werden muss, welches Resultat für den jeweiligen Datensatz auf basis der Gesamtinformationen am zutreffendsten ist.

Die fuzzifizierten Datensätze müssen miteinander vergleichbar sein und es werden Methoden benötigt, die anhand mehrerer Datensätzen die wahrscheinlichste Zuordnung vornehmen können und somit eine größere Toleranz gegenüber unscharfer Daten gewährleisten.

4.2.2 Navigation

Die Navigation innerhalb des Flowmodells basiert auf den Zuständen der Aktivitäten und die Datensätzen die von dem Kontextsystem geliefert und anschließend für die weitere Verarbeitung aufbereitet wurden. Hier werden für jeden der in subsection 3.2.3 definierten Zustände Abläufe und Methoden definiert die pro Aktivität ausgeführt werden.

In subsection 4.1.2 wurde eine Übersicht gegeben, wie sich der Zustand des gesamten Flowmodells während der Ausführung ergibt und das zwischen drei Teilen unterschieden werden kann (siehe Abbildung 4.2).
Die Aktivitäten, die sich in dem inaktiven Bereich befinden, werden während der Ausführung zu diesem Zeitpunkt von der Navigation noch nicht berücksichtigt. Die Aktivitäten, die sich in den Zuständen *Prepare*, *Active* oder *Can Complete* befinden werden aktuell abgearbeitet und bilden somit den Bereich Flowmodells in dem aktuell die eintreffenden Kontextinformationen ausgeliefert werden und basierend darauf Navigationsentscheidungen getroffen werden. Dieser Bereich kann auch als unsicherer Bereich beschrieben werden, da hier nur eine vorläufige Zuordnung der Datensätze vorgenommen wurde und Navigationsentscheidungen getroffen wurden, die aufgrund geänderter Informationen revidiert und neu evaluiert werden kann. In dem Abgeschlossenen Bereich befinden sich die Aktivitäten, die in dem Zustand *Complete* sind, in diesem Bereich ist die Navigation abgeschlossen und durchgeführte Datenzuordnungen und Navigationsentscheidungen können nicht mehr geändert werden. Dieser Bereich kann als stabil beschrieben werden, da hier alle Aktivitäten mit mindestens der geforderten Sicherheit korrekt durchgeführt wurden.

Die Startaktivität wird dadurch definiert, dass sie keine eingehenden Kanten besitzt, während der Initialisierung wird diese Aktiviät in den Zustand *Active* versetzt und für den Empfang von Eventdaten registriert. Ab diesem Zeitpunkt werden die Aktivitäten nur noch anhand ihres Zustands bearbeitet.

In jedem Navigationsschritt wird für alle Aktivitäten innerhalb des aktiven Bereichs geprüft ob eine Aktion ausgeführt werden kann. Abhängig vom aktuellen Zustand der Aktivität können das die Folgenden sein:

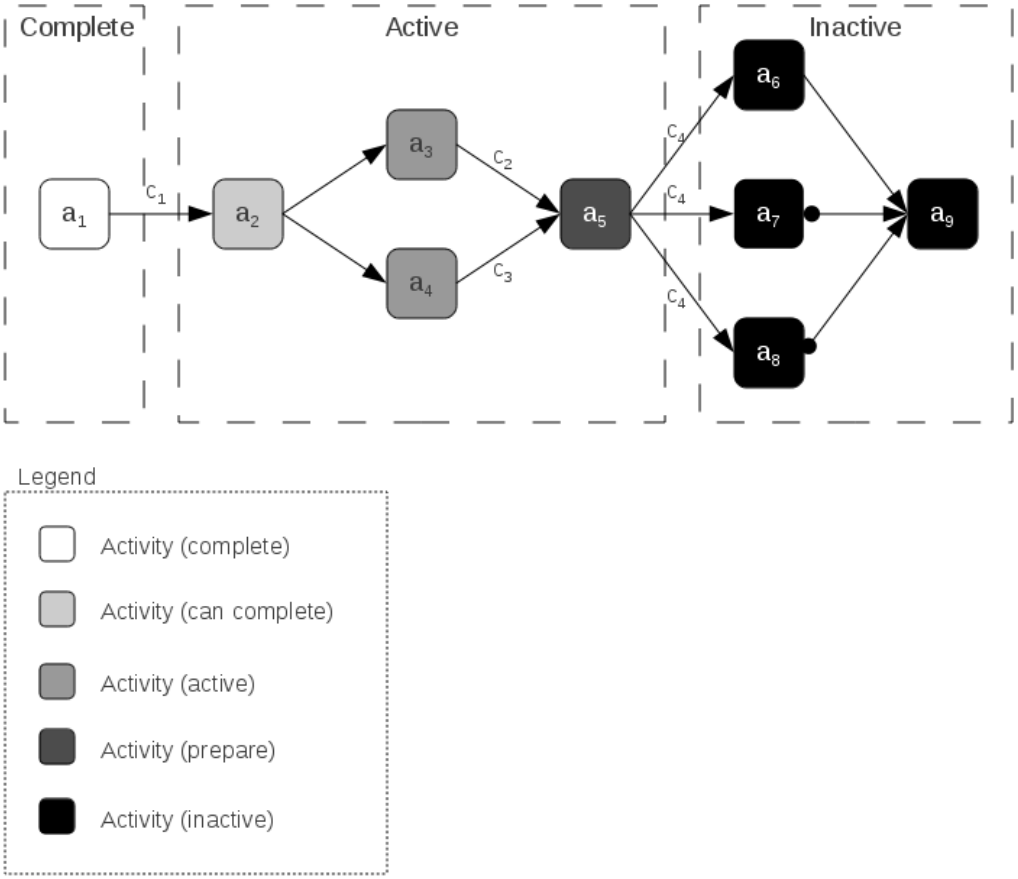

Abbildung 4.2: Flowmodell Zustand

Active - Aktivitäten in diesem Zustand werden über geliefert Eventdaten informiert. Die Registrierung für den Empfang wird während des Zustandsübergangs vorgenommen. Zunächst wird für eine Aktivität in diesen Zustand geprüft, ob für die nachfolgenden Aktivitäten der korrekte Zustand gesetzt ist, das ist vor allem bei den Zustandsübergängen wichtig. Es werden alle ausgehenden Transitionen der Aktivität geprüft und wenn sich eine der folgenden Aktivitäten in dem Zustand *Inactive* befindet wird diese in den Zustand *Prepare* versetzt.
Im Folgenden wird der Zustand der Aktivität geprüft und ermittelt, ob basierend auf dem aktuellen Datebestand eine Zustandsänderung vorgenommen werden kann. Im Ersten Schritt wird dabei geprüft, ob für jeden regisitrierten Event-Typ mindestens ein Event empfangen wurde. Ist das der Fall, wird für die empfangenen Datensätze die Evaluierung angestoßen (sieh Abschnitt 4.3.2), dabei wird ermittelt, mit welcher Zuverlässigkeit die erforderlichen Aktionen ausgeführt wurden. Ergibt diese Evaluation, dass alle für die Aktvität benötigten Nutzeraktionen mit der notwendigen Sicherheit augeführt wurden wird die Aktivität in den Zustand *Can Complete* versetzt.
Innerhalb des Flows kann es Aktivitäten geben, die als Entscheidungspunkte innerhalb des Flows fungieren und basierend auf dem aktuellen Zustand den weiter zu verfolgenden Pfad ermitteln. Für diese Entscheidungen werden den ausgehenden Transitionen Bedingungen angeheftet, die für die Aktivierung der entsprechenden Transition erfüllt werden müssen. So können für zwei ausgehenden kanten unterschiedliche Events gefordert werden, die währeden der Abarbeitung der Aktivität emfpangen werden müssen. Es kann in so einem Falls nicht nur komplett für eine Aktivität der Empfang aller benötigten Events abgefragt und evaluiert werden, sondern auch getrennt nach ausgehenden Transitionen. So kann basierend auf den empfangenen Events die Zuverlässigkeit ermittelt werden, dass die Aktionen ausgeführut wurden, die eine bestimmte Entscheidung nach sich ziehen.

Prepare - Dieser Zustand wird wie in Abschnitt 4.1.1 beschrieben, eingeführt um die Robustheit gegenüber einer unsicheren Datenversorgung eingeführt und erfüllt insbesondere den Zweck während der Ausführung fehlende Eventdaten oder Datensätze, die nicht die benötigte Sicherheit aufweisen zu tolerieren. Eine Aktivität wird in den Zustand *Prepare* gesetzt, wenn eine direkte Vorgängeraktivität in den Zustand *Active* gesetzt wird. Beim setzten des Zustands wird für diese Aktivität geprüft welche Event-Typen erwartet werden und sie wird dementsprechend für den Empfang dieser Events registiert.
Wie Aktivitäten des Zustand *Active* werden auch diese Aktivitäten mit eingehenden Events versorgt, für die sie registriert wurden. Das erfüllt den Zweck, dass die Aktivität bereits mit Informationen versorgt wird, bevor die vorausgegangene Aktivität abgeschlossen wurde. Unter der Annahme, dass eine Aktivität nicht abgeschlossen werden kann, da ein Event nicht erfasst oder nicht die erforderliche Sicherheit aufweist und die Ausführung des Flows in so einem Fall nicht weiter durchgeführt werden könnte, kann mit dieser Vorkehrung geprüft werden ob der folgende Schritt, hier die Aktivität im Zustand *Prepare* korrekt ausgeführt wurde und so angenommen werden kann, dass die vorherige Aktivität aufgrund der unsicheren Datenversorgung nicht korrekt abgeschlossen werden konnte.

Ähnlich wie in dem Zustand *Active* werden für diese Aktivitäten zunächst geprüft, ob für jeden registrierten Event Typ mindestens ein Event empfangen wurde. Ist dies der Fall werden die empfangenen Events analysiert und geprüft ob sie die Anforderunge der entsprechenden Aktivität erfüllen. Das entspricht einer isolierten Prüfung der einzelnen Aktivität ohne Berücksichtigung des Gesamtzustands des Flows.
Tritt der Fall ein, dass die Aktivität basierend auf den emfpangenen Daten abgeschlossen werden könnte, muss entschieden werden, ob diese Aktivität zum aktuellen Zeitpunkt als korrekt ausgführt gekennzeichnet werden darf. Ein einfaches Entscheidungskriterium dafür ist, die geforderte Zuverlässigkeit der Daten zu erhöhen und so eine weiter Ausführung des Flows nur dann zu ermöglichen, wenn verlgeichsweise sicher davon ausgegangen werden kann, dass die erfassten Informationen die Ausführung dieser Aktion identifizieren. Zusätzlich zur Steigerung der geforderten Zuverlässigkeit kann analysiert werden welche Anforderungen in der vorausgegangen Aktivität noch nicht erfüllt wurde, z.B. die Anzahl der Events die für einen Abschluss noch benötigt werden. Damit können weitere Regeln definiert werden, unter welchen Umständen eine Aktivität innerhalb des Flows nicht vollständig ausgführt sein muss, indem angegeben werden kann wie viele Datensätze maximal fehlen dürfen und basieren darauf, welche zusätzlichen Anforderungen die folgenden Aktivität erfüllen muss um das auszugleichen.
Wie im Zustand *Active* wird zunächst ermittelt, ob alle benötigten Daten empfangen wurden und die geforderte Zuverlässigkeit aufweisen und ob es sich bei der Aktivität um einen Entscheidungspunkt handelt, bei dem die Anforderungen nicht für die gesamte Aktivität, sondern in Abhängigkeit der ausgehenden Transitionen ermittelt werden muss. Sind diese Anforderungen erfüllt, muss geprüft werden ob aufgrund des Zustands zusätzlich gestellte Bedingungen wie die Forderung nach einer höheren Zuverlässigkeit erfüllt werden können. Ist dies der Fall, kann die Aktivität auf den Zustand *Can Complete* gestezt werden und die nachfolgenden Aktivitäten werden entsprechend aktiviert.

Can Complete - Aktivitäten in diesem Zustand haben für jedes benötigte Aktion ein entsprechendes Event emfpangen, dass die Anforderungen erfüllt und können abgeschlossen werden, wenn weitere Voraussetzungen für die folgenden Aktivitäten erfüllt werden. Da sich diese Aktivitäten noch in dem unsicheren Bereich des Flowmodells befinden und die Zuordnung der Eventdaten geändert werden kann, werden sie weiterhin mit eingehenden Informationen versorgt.
Die Anforderung, die erfüllt werden muss, dass die Aktivität auf den Zustand *Complete* versetzt werden kann ist, dass sich alle folgenden Aktivitäten ebenfalls im Zustand *Can Complete* befinden. Das bedeutet, dass die folgenden Aktivitäten ebenfalls für alle Aktionen ein Event mit ausreichender Sicherheit emfpangen haben und keine Konflikte zwischen den diesen Aktivitäten bei der Zuordnug der Events besteht. In diesem Fall kann angenommen werden, dass die Zuordnung der Events korrekt ist, keine weiteren Änderungen auftreten und die Aktivität in einen stabilen Zustand überführt werden kann.
Wird nach der Prüfung ob eine Aktivität in den Zustand *Complete* versetzt werden kann entschieden, dass die Voraussetzungen dafür erfüllt sind, werden die zugeordneten Events

konsumiert, sowie die Registrierungen für die weitere Eventversorgung gelöscht.
Besonders behandelt müssen in diesem Zustand die Endaktivitäten, da hier nicht mehr der Zustand der folgenden Aktvitäte geprüft werden kann. Eine Endaktivität wird dadurch definiert, dass sie keine ausgehenden Transitionen besitzt. In diesen Fall kann die Aktivität auf den Zustand *Complete* gesetzt werden und dadurch wird ebenfalls signalisiert, dass der Flow erfolgreich durchgeführt wurde und abgeschlossen werden kann.

Complete - Dies ist der finale Zustand der Aktivitäten und wird gesetzt, wenn alle Eventdaten empfangen und der Aktivität zugeordnet werden konnten. Bevor eine Aktivität in diesen Zustand versetzt wird, wird wie oben beschrieben geprüft, ob den folgenden Aktivitäten ebenfalls alle benötigten Events zugeordnet werden konnten, so dass davon ausgeganen wird, dass hier keine Änderung der Zuordnung vorgenommen werden muss.
Die Aktivitäten in diesem Zustand werden nicht mehr weiter mit eingehenden Events versorgt und im weiteren Verlauf nicht mehr von der Navigation berücksichtigt.

Inactive - Der initiale Zustand aller Aktivitäten ist *Inactive*. Die Startaktivitäten werden während der Initialisierung des Flowmodells aktiviert. Während der Ausführung werden diese Aktivitäten nicht mit eingehenden Informationen versorgt und innerhalb der Navigation nicht berücksichtigt. Erst wenn sie aufgrund einer Zustandsänderung einer vorausgegangen Aktivität in den Zustand *Active* versetzt werden sie innerhalb der Navigationsschritte abgearbeitet.

4.3 Komponenten

Die Funktionalität des Systems wird in verschiedene, nach ihrer Funktion gruppierten Komponenten aufgeteilt. In Abbildung 4.3 ist eine solche Aufteilung als Diagramm gegeben. Das System gliedert sich in vier Hauptkomponenten, die im Folgenden erläutert werden: die Komponente zur Fuzzifizierung der eingehenden Kontextinformationen (Abschnitt 4.3.1), die Komponente zur Evaluation der Empfangenen Datensätze und Determinierung welche die gestellten Anforderungen erfüllten (Abschnitt 4.3.2), die Komponente die die emfpangenen Eventdaten verwaltet, die Zuordnung zu den Aktivitäten übernimmt und Konflikte in der Datenverteilung behandelt (Abschnitt 4.3.3 und die Komponente zur Navigation innerhalb des Flowmodells, die basieren auf den empfangenden Eventdaten und der aktuellen Zuordnung dieser Daten die Entscheidungen über den nächsten auszuführenden Navigationssschritt trifft.

4.3.1 Fuzzifizierer

In Abschnitt 3.5.1 wurde die Notwendigkeit beschrieben, die von dem Kontextsystem gelieferten Daten aufzubereiten und in ein Format zu bringen das für die Zuordnung der

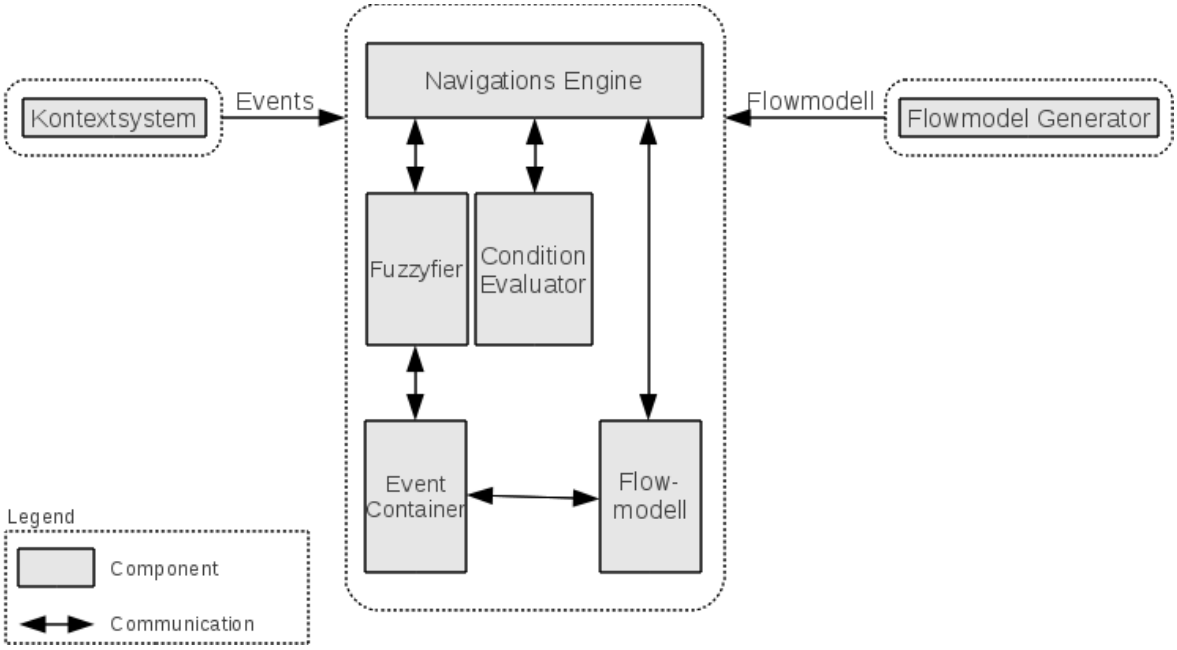

Abbildung 4.3: Architektur

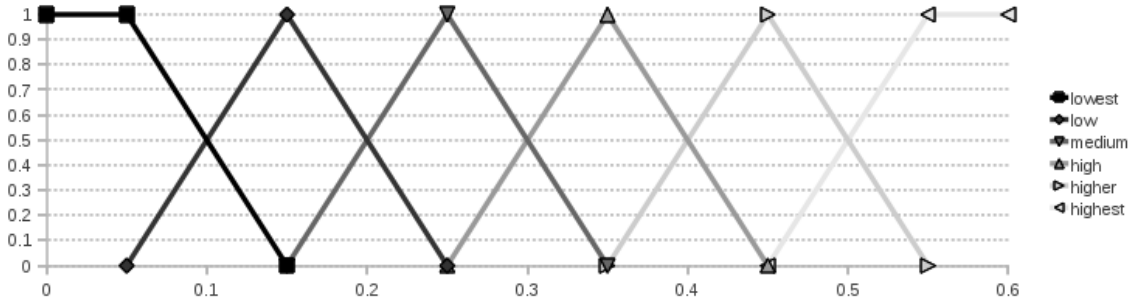

Abbildung 4.4: Abbildungsfunktion

Datensätze und die Navigation verwendet werden kann.
Die erfassten Daten, die eine Wahrscheinlichkeitverteilung für eine erkannte Situation liefert werden mittels einer Funktion (Abbildung 4.4) auf eine Fuzzy Menge abgebildet, die in Form von linguistischen Variablen die Sicherheit, dass eine bestimmte Situaion erfasst wurde definiert.

4.3.2 Condition Evaluator

Die Events, die von dem Kontextsystem geliefert werden, werden im ersten Schritt nur über die definierten Event Typen an die registrierten Aktivitäten verteilt. Wenn eine Aktivität zu jeden benötigten Typ mindestens ein Event empfangen hat, wird die Liste der empfangenen Events dem *Condition Evaluator* übergeben und analysiert welche der verfügbaren Events die geforderten Bedinungen erfüllen.
Die zu erfüllenden Bedingungen sind zugehörig zu den ausgehenden Transitionen einer Aktivität definiert, so kann bei einer Aktivität entschieden werden, für welche Transition die Bedinungen erfüllt wurden und so eine Navigationsentscheidung im Bezug auf den weiteren Verlauf getroffen werden.
Eine Bedinung kann den Empfang von einem oder mehreren Events erfordern. Innerhalb des *Condition Evaluators* werden die geforderten Bedinungen ausgewertet und überprüft, ob die vorhandenen Daten das geforderte Mindestmaß an zuverlässigkeit aufweisen.
Für die Erfüllung können unterschiedliche Forderungen gestellt werden. Wie in Abschnitt 4.2.2 aufgeführt werden dabei in Abhängigkeit von dem Zustand der jeweiligen Aktivität Ergebnisse definiert, die erfüllt werden müssen.
Basierend auf der Datenaufbereitung durch den Fuzzifizierer können hier Regeln erstellt werden die für eine Aktivität, die drei Events benötigt beispielsweise definiert, dass mindestens zwei der emfpangenen Events die erforderliche Situation mit *hoher Sicherheit* erkannt haben und für die dritte eine *mittlere Sicherheit* ausreicht.

4.3.3 EventContainer

Der *EventContainer* ist die zentrale Komponente für die Eventdatenhaltung und -verteilung. Alle von dem Kontextsystem erfassten Informationen werden in Form von Events nach der Aufbereitung durch den *Fuzzifizierer* in dem *EventContainer* abgelegt und in Abhängigkeit der jeweiligen Event Typen organisiert.
Die Aktivitäten werden bei ihrer Aktivierung (vgl. 4.2.2) für den Empfang der benötigten Event Typen bei dem *EventContainer* registriert. Beim Empfang eines neuen Events, werden alle für den entsprechenden Event Typ registrierte Aktivitäten über das neue Event informiert.

Der *EventContainer* ist ebenfalls für die Zuordnung der einzelnen Events zu den Aktivitäten zuständig. Aus den von dem *Condition Evaluator* ermittelten Events, die die geforderten Bedingungen erfüllen, werden von der Aktivität bei dem Event Container angefragt. Für die angefragten Events wird zunächst überprüft, welche noch verfügbar, also noch keiner anderen Aktivität zugeordnet wurden und im Folgenden das Event der der Aktivität zugeordnet, das für die Erfüllung der Bedingungen die höchste Sicherheit aufweist.
Im Analyseteil wurde ein Problem geschildert, das aufgrund der unsicheren Datenerfassung

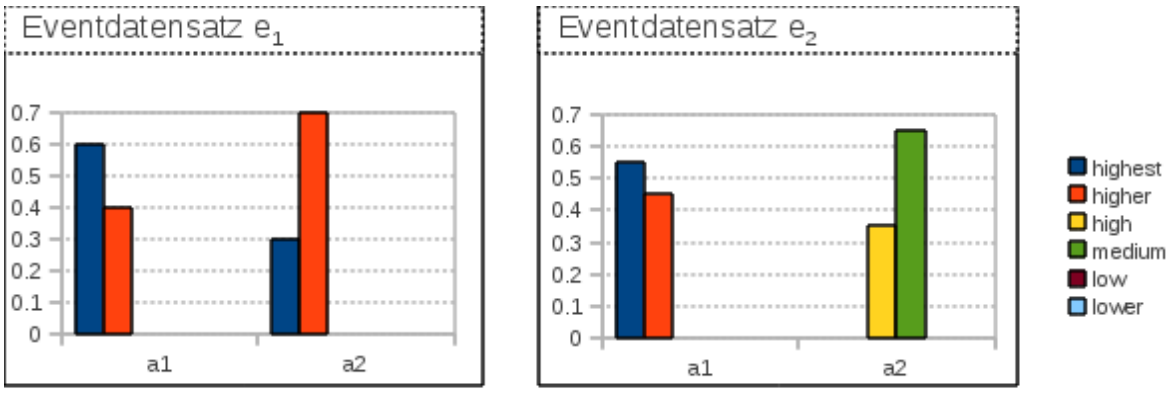

Abbildung 4.5: Fuzzydatensätze der Events e_1 und e_2

dazu führen kann, dass Datensätze fehlerhaft interpretiert und demzufolge nicht der korrekten Aktivität zugeornet wurden. Im Folgenden wird dieser Problemfall aufgegriffen und Möglichkeiten dargestellt, wie sie innerhalb des Event Containers behandelt und aufgelöst werden können.
Vorausgesetzt die Kontextinformationen sind zuverlässig und es treten keine Fehler in der Datenerfassung auf würde jedes benötigte Event exakt erfasst werden können und für jede Aktivität würde nur genau dieses Event die geforderten Bedinungen erfüllen. In so einem Fall würde die Anfrage der Aktivitäten nach den benötigten Events an den Event Container pro Event Typ genau ein Event beinhalten, das dieser Aktivität zugeordnet werden kann.
Durch die unsicheren Daten entsteht das Problem, dass das nicht der Fall ist und für einen erfassten Datensatz die wahrscheinlichste Interpretation angenommen wird. Unter diesen Voraussetzungen kann es dazu kommen, dass für ein benötigtes Event einer Aktivität mehrere Events die erforderliche Sicherheit erfüllen. Dieses Problem entsteht dadurch, dass bei der Analyse eines Eventdatensatzes die Situation mit der höchsten Zuverlässigkeit angenommen wird unabhängig davon, ob die in diesem Fall angenommene Situation bereits für einen anderen Datensatz angenommen wurde.
Die Aufgabe des *EventContainers* hier besteht darin, basierend auf einem umfangreicheren Informationsbestand zu ermitteln in welchem Fall die Interpretation eines Datensatzes angepasst werden muss.

Angenommen die Events e_1 und e_2 sind vom selben Event Typ mit folgenden Daten (siehe Abbildung 4.5) in dem Event Container verfügbar. Während der Evaluation für Aktivität a_1 wurden beide Events als passende Kandidaten, die die Bedingungen erfüllen ermittelt. So werden in der Anfrage für diesen Event Typ beide Events übergeben. In diesem Fall, wenn noch keines der angefragten Events einer Aktivität zugeordnet wurde, ermittelt der EventContainer das Event mit der höchsten Sicherheit und weist es der anfragenden Aktivität zu. In diesem Fall ist das das Event e_1.

Werden die erhaltenen Events für die Aktivität a_2 evaluiert und nur das Event e_1 erfüllt die Bedinungen wird auch nur dieses Event für den entsprechenden Typ beim Event Container angefragt. Hier tritt das Problem auf, dass das angefragte Event bereits einer anderen Aktivittät zugeordnet wurde und somit nicht mehr verfügbar ist. In so einem Konfliktfall werden die die aktuelle Anfrage der Aktivität a_2 mit der Anfrage der Aktivität verglichen, der das angefragte Event zugeordnet wurde, in diesem Fall a_1. Bei dem Vergleich der beiden Anfragen kann festgestellt werden, dass für die Aktivität a_1 zwei Events angefragt wurden, die die geforderten Bedingungen erfüllen. In diesem Fall kann die Zuordnung aufgehoben werden und das Event der Aktivität a_2 zugeordnet werden, der Aktivität a_1 wird im Folgenden das Event e_2 zugeordnet.
Ein weiterer Problemfall für die Eventzuordnug ist, wenn ein Event von zwei Aktivitäten angefordert wird und für keine Aktivität wie im vorherigen Beispiel eine Alternative verfügbar ist. Trivial könnte so verfahren werden, dass wenn keine Alternativen bestehen das Event der Aktivität zugeordnet bleibt, die die Anfrage zuerst gestellt hat und somit keine erneute Zuordnung durchgeführt werden muss. Problematisch ist das in dem Fall, wenn eine Aktivität für die korrekte Ausführung des Flows notwendig und die andere optional ist. Diese Voraussetzung muss bei der Zuordnung der Events berücksichtig werden. Wenn keine Alternativen bestehen, werden bei der Verteilung der Events die Aktivitäten bevorzugt, die *mandatory* sind.

4.3.4 Navigationsengine

Die *Navigationsengine* ist die zentrale Komponente für die Steuerung der Abläufe und Zustände. Hier werden die die Zustände der Aktivitäten überwacht, die entsprechenden Aktionen angestoßen und die Koordination zwischen den beteiligten Komponenten durchgeführt.
In Abschnitt 4.2.2 wurde die Zustandsbasierte Navigation bereits eingeführt und die Bedeutung der Zustände, die eine Aktivität annehmen kann, erläutert.

Die *Navigationsengine* prüft für jede Aktivität innerhalb des aktiven Bereichs (vgl. Abschnitt 4.1.2) welche Aktionen ausgeführt werden müssen und ob der Zustand der Aktivität geändert werden kann. Dabei werden die folgenden Schritte ausgeführt. Kann ein Schritt dabei nicht vollständig ausgeführt werden oder führt zu keinem positiven Ergbnis wird die Abarbeitung für die entsprechende Aktivität unterbrochen und mit der nächsten fortgefahren.

- Es wird geprüft, ob für jeden registrierten Event Typ mindestens ein Event empfangen wurde.
- Der Condition Evaluator (subsection 4.3.2) prüft, ob für jeden Typ mindestens eines der emfpangenen Events die Bedingungen erfüllt.

- Die Events, die die Bedingungen erfüllen werden dem Event Container (4.3.3) übergeben, der versucht die angefragten Events der Aktivität zuzuordnen.
- Können die Events zugeordnet werden, kann die Aktivität in den Zustand *Can Complete* versetzt werden und dementsprechend die Zustände der folgenden Aktivitäten gesetzt werden.

Weitere Vorkehrungen sind für Aktivitäten in dem Zustand *Prepare* erforderlich, da hier zusätzliche Bedingungen erfüllt werden müssen, die für die jeweilige Ausführung eines Flows definiert werden können.
Es kann für diesen Fall eine höhere Zuverlässigkeit der empfangenen Events gefordert werden oder dass der Vorgängeraktivität, sofern sie sich nicht in dem Zustand *Can Complete* befindet, nur eine maximale Anzahl benötigter Events fehlen darf.

5 Evaluation

In diesem Kapitel wird die Leistungsfähigkeit des entwickelten Algorithmus untersucht und die Ergebnisse diskutiert.
In section 5.1 wird das Flowmodell vorgestellt, das für die Testläufe verwendet wird. In section 5.2 werden zu erwartende Probleme, die in der Analyse identifiziert wurden aufgeführt. In section 5.3 werden die Konfigurationsparameter für die Testläufe erklärt und die daraus folgenden Erwartungen dargestellt. In section 5.4 werden die Ergebnisse der Testläufe dargestellt und die Ergebnisse diskutiert und mit den Erwartungen verglichen.

5.1 Testmodell

5.2 Problemstellung

Ausgehend von einer Situation, dass alle Nutzeraktionen korrekt und sicher erfasst werden können werden für die Testläufe folgende Probleme innerhalb der Datenerfassung simuliert, die die unsichere Datenerfassung in der Realität wiederspiegeln sollen.

Rauschen - Von einem Ausgangswert, bei dem eine Nutzeraktion zu 100% Zuverlässigkeit erkannt wurde, wird entsprechend der Testkonfiguration ein Prozentsatz abgezogen und auf weitere Nutzeraktionen verteilt.

Events löschen - Aus der für den Simulator generierten Eventliste die über das Kontextsystem während der Ausführung ausgeliefert werden sollen wird eine bestimmte Anzahl Events gelöscht um Situationen zu simulieren, in den einzelne Nutzeraktionen nicht erfasst werden konnten.

Reihenfolge ändern - Zudem werden innerhalb der Eventliste einzelne Events zufällig ausgewählt und um eine maximale Anzahl von Positionen in der Liste verschobem, um die Reihenfolge innerhalb der Eventversorung zu beeinflussten und diesbezüglich die Robustheit des Algorithmus zu evaluieren.

Events generieren - Analog zum Löschen von Events aus der Eventliste werden Events mit zufälligen Werten hinzugefügt. Dadurch soll die fehlerhafte Interpretation von Umgebungsrauschen simuliert werden.

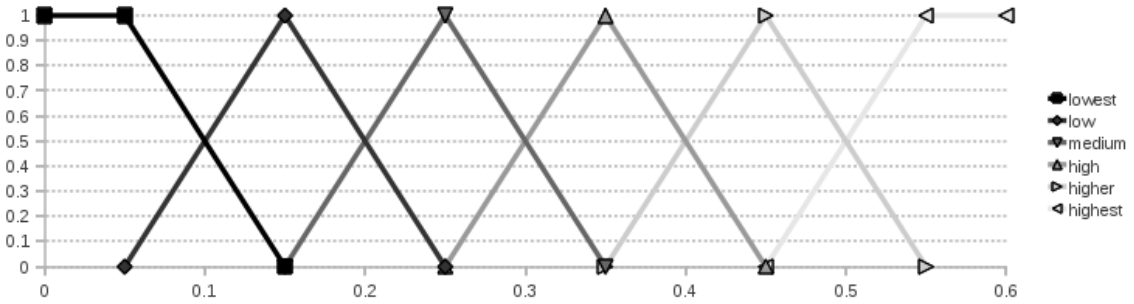

Abbildung 5.1: Fuzzyfunktion

5.3 Testkonfiguration

Für die in Abschnitt 4.3 beschriebenen Komponenten ist es möglich, Konfigurationen vorzunehmen, die die Ausführung des Flows beeinflussen. Zunächst werden die verfügbaren Konfigurationsparameter aufgeführt und angegeben innerhalb welchen Bereichs diese für die Testläufe konfiguriert werden. Anschließen werden die konkreten Konfigurationen für die durchgeführten Testläufe aufgeführt.

Für den *Fuzzifizierer* wird die in Abbildung 5.1 dargestellte Funktion verwendet. Von dem Kontextsystem werden die Eventdaten mit zugehörigen Zuverlässigkeitswerten geliefert, dass eine bestimmte Situation erfasst wurde. Diese Werte sind als Eingabewerte auf der x-Achse abgetragen. Die y-Achse definiert für einen gegebenen Eingabewert die Zugehörigkeit zu den definierten linguistischen Sicherheitsvariablen.
Die Abbildungsfunktion wird für die verschiedenen Testläufe nicht geändert.

Der *Condition Evaluator* definiert die Bedinungen, die ein empfangenes Event erfüllen muss, um einer Aktivität zugeordnet werden zu können. Diese Evaluation basiert auf den fuzzifierzierte Eventdatensätzen und demtensprechen kann hier angegeben werden, welche Zugehörigkeitwerte zu den definierten Sicherheitsvariablen erfüllt werden müssen.
Bei den Testläufen wird für die Zuornung eines Events zu einer Aktivität die Sicherheitsvariable *highest* gefordert.

In der *Navigationsengine* kann konfiguriert werden, wie viele erwartete Events bei einer Aktivität maximal fehlen dürfen, dass die folgende Aktivität von dem *Prepare* in den *Can Complete* Zustand wechseln darf. Damit kann gesteuert werden wie viele Events, die nicht korrekt erfasst wurden oder die benötigte Zuverlässigkeit aufweisen bei der Ausführung toleriert werden können. Für die Testläufe wird hier maximal ein fehlendes Event zugelassen.

Das *Kontextsystem* ist für die Auslieferung der Events an die *Navigationsengine* verantwortlich. Dabei wird zunächst eine Liste von Events erstellt, die die korrekte Ausfühung des Flowmodells ermöglicht und für jedes Event eine Sicherheit von 100% für die korrekte Erfassung

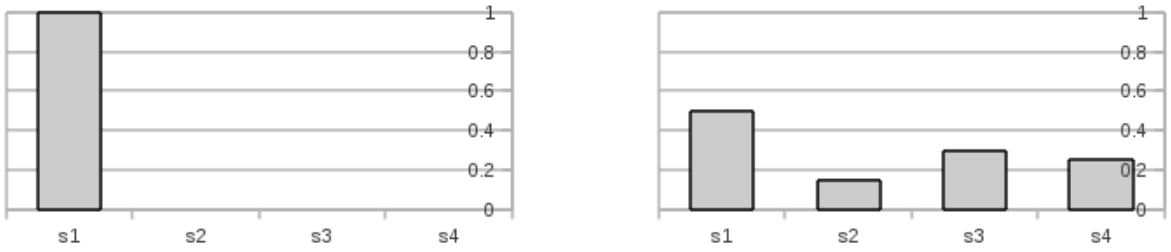

Abbildung 5.2: Eventdaten

liefert (siehe Abbildung 5.2 links).
Für die Testläufe wird diese Eventliste sowie die Datensätze der einzelnen Events verändert, um die Unsicherheit in der Erfassung der Kontextinformationen zu simulieren.
Für jeden Durchgang wird die Sicherheit für die korrekte Erfassung einer Situation innerhalb des Events auf einen bestimmten Basiswert zwischen 60% und 30% herabgesetzt, dabei wird dieser Wert schrittweise um 5% gesenkt. Zusätzlich wird eine Variationsbereich von 5% bis 10% definiert, aus welchem pro Event zufällig ein Wert ausgewählt wird, der der Sicherheit hinzugefügt oder abgezogen wird.
Um das Rauschen und Umgebungseinflüsse einer realen Situaion zu simulieren, werden die Werte für die Erfassung einer anderen Situation innerhalb des Events zufällig vergeben (vgl. Abbildung 5.2 rechts).

Neben der Simulation von Unschärfe in der Datenerfassung der einzelnen Events können weitere Änderungen an der Eventliste konfiguriert werden. Es kann der prozentuale Anteil der Events definiert werden, der aus der Eventliste gelöscht werden soll und so den Anteil der nicht erfassten Events simuliert. Zudem kann die Reihenfolge der Events verändert werden, dazu kann die Anzahl der Events und der Abstand innerhalb der Reihenfolge angegeben werden wieviele Events um maximal wieviel Positionen in der Eventliste verschoben werden können.
Für die Evaluation werden bis zu 20% der Events aus der Liste gelöscht und bis zu 10% der Events um maximal 4 Positionen in der Liste verschoben.

Ähnlich dem Löschen kann eine Anzahl Events definiert werden, die der Eventliste hinzugefügt werden. Diese Events werden mit zufälligen Werten befüllt und sollen die fehlerhafte Interpretation von Umgebungsrauschen als eingetretenen Situation simulieren. Für die Testläufe werden hier bis zu 10% der in der Eventliste verfügbaren Events generiert und an zufälligen Positionen eingeordnet.

Basierend auf diesen Konfigurationsmöglichkeiten wurden folgende Gruppen von Testläufen konfiguriert und ausgeführt, dabei wird jede Konfiguration 100 mal mit den selben Parametern ausgeführt um eine statistische Signifikanz zu erreichen.
Innerhalb einer Gruppe werden folgende Werte fest definiert.

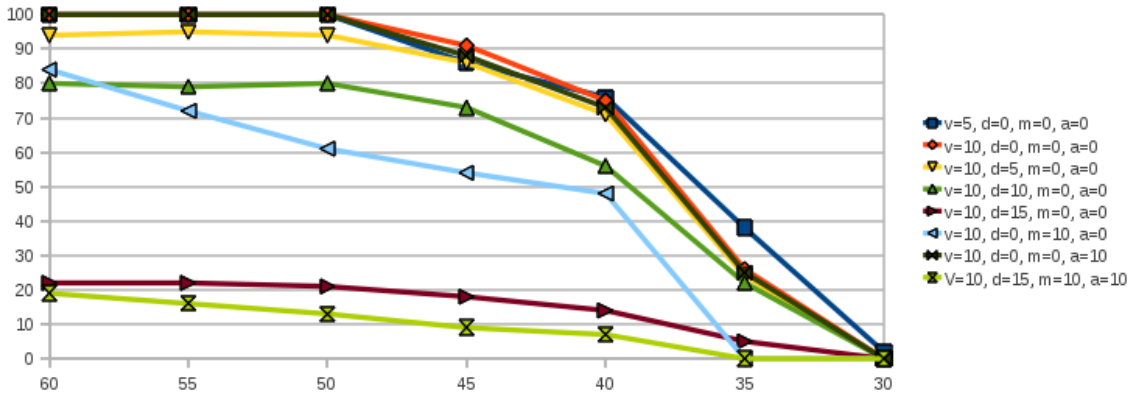

Abbildung 5.3: Simulations Ergebnisse

- Die Varianz für die Sicherheit einer korrekten Erfassung.
- Prozentsatz der zu löschenden Events.
- Prozentsatz der Events, die innerhalb der Reihenfolge verschoben werden sollen.
- Prozentsatz der zusätzlich eingefügten Events.

Mit diesen Parametern werden Testläufe durchgeführt bei denen der Basiswert für die korrekte Erfassung einer Situation in 5% Schritten von 60% auf 30% herabgesetzt wird. Für die Gruppen werden folgende Parameter definiert.

1. Varianz = 5%, löschen = 0%, verschieben = 0%, zuätzliche Events = 0%
2. Varianz = 10%, löschen = 0%, verschieben = 0%, zuätzliche Events = 0%
3. Varianz = 10%, löschen = 5%, verschieben = 0%, zuätzliche Events = 0%
4. Varianz = 10%, löschen = 10%, verschieben = 0%, zuätzliche Events = 0%
5. Varianz = 10%, löschen = 15%, verschieben = 0%, zuätzliche Events = 0%
6. Varianz = 10%, löschen = 0%, verschieben = 10%, zuätzliche Events = 0%
7. Varianz = 10%, löschen = 0%, verschieben = 0%, zuätzliche Events = 10%
8. Varianz = 10%, löschen = 15%, verschieben = 10%, zuätzliche Events = 10%

5.4 Analyse

Bei der Simulation wurde die Anzahl der erfolgreich abgeschlossenen Flows gemessen. Diese Zahl kann als als Robustheit des Algorithmus interpretiert werden, in dem die Werte anhand der jeweiligen Konfiguration gegenüber gestellt werden.

Sicherheit einer Korrekten Erfassung - Bei der Schrittweise verringerten Sicherheit einer korrekten Erfassung ist die Erfolgsquote oberhalb der 50% Grenze sehr hoch. Bis 40% fällt sie leicht ab, bleibt aber in den ersten zwei Testläufen noch über 75%. Unterhalb eines Basiswerts von 40% stürzt der Anteil der Erfolgreich abgeschlossenen Flows aber regelrecht ab, so dass bei 30%, außer im ersten Testlauf, keine erfolgreiche Durchführung mehr möglich ist.

Wenn die Sicherheit, dass eine Situation korrekt erfasst wurde nach der Datenaufbereitung der Fuzzy Menge *highest* zugeordnet ist, kann das Event der entsprechenden direkt Aktivität zugeordnet werden. Daher ist es nicht überraschend, dass bei den ersten Testläufen, bei den die Eventliste nicht manipuliert wurde eine Erfolgsquote von 100% erreichet wurde.
Ab einem Wert von 45% Sicherheit für eine korrekte Erfassung, fällt das Event aus der höchsten Stufe heraus und bekommt nur noch den Wert *higher* zugeordnet. Diese Zuordnung wird unterhalb von einem Basiswert von 40% nicht mehr vorgenommen und der Wert fällt noch eine Stufe ab. Diese Einstellung lässt sich sehr deutlich an den Ergebnissen ablesen.
Damit erfüllt die Simulation in diesem Bereich die Erwartungen, da eine höhere Erfolgsquote mit der gewählten Testkonfiguration nicht zu erwarten war.

Events löschen - In den Testläufen 3 und 4, in den bis zu 10% der Events gelöscht wurden ist die Anzahl der Erfolgreichen Ausführungen noch relativ gut. Bei 10% gelöschter Events sinkt die Quote um ca. 20% im Durchschnitt. Im 5. Testlauf, in dem 15% der Events gelöscht wurden fällt das Ergebnis allerdings sehr viel schlechter aus. Selbst mit einem hohen Basiswert für die korrekte Erfassung der Events können nur 22% der Flows Abgeschlossen werden.

Die Toleranz gegenüber fehlender Events oder unzureichender Sicherheit bei der Erfassung ist gut. Bei 5% fehlender Events fällt die Erfolgsquote im Schnitt um 5%, wird die Menge allerdings auf 10% erhöht, ist es schon ein Abfall um 20% und bei 15% werden bei einer sehr hohen Zuverlässigkeit nur 22% der Flows erfolgreich abgeschlossen.
Die Ursache dafür, dass die Werte so stark abfallen lässt sich einfach erklären. Der Algorithmus toleriert ein fehlendes Event pro Aktivität, aber nur vorausgesetzten Falls, dass die folgenden Aktivität alle Events empfangen hat und abgeschlossen werden kann. Bei dem Flowmodell, das für die Simulation verwendet wurde erwarten die Aktivitäten im Durchschnitt 3 Events. Es darf also bei einer gleichmäßigen Verteilung der fehlenden Events nur jedes sechste Event fehlen.

Events verschieben - Das Verschieben von Events und somit die Beeinflussung der Reihenfolge hat eine noch stärkere Auswirkung auf das Ergebnis, als das Löschen der gleichen Anzahl von Events. Bis 40% sinkt die Erfolgsquote auf 48% und fällt anschließen stark ab.

Das Verändern der Reihenfolge hat einen größeren Einfluss als erwartet, zu erklären ist das in den Bereichen mit einer geringeren Zuverlässigkeit damit, dass durch eine fehlerhafte Ordnung der Events die Wahrscheinlichkeit erhöht wird, dass ein Event einer falschen Aktivität zugeordnet wurde. Die Auflösung so eines Konflikts ist ungleich schwerer, wenn die die korrekte Zuordnung nicht mehr einfach erkannt werden kann und eventuell auch im Vergleich von zwei Events nicht festgestellt werden kann, welche Interpretation in diesem Fall korrekt ist.

Einfügen von Events - Das einfügen von Events mit zufälligen Daten hat fast keine Auswirkung auf die Anzahl der abgeschlossenen Flows. Das ist dadurch erklärbar, dass die mit zufälligen Werten bestückten Events vielleicht als mögliche Kandidaten für die Aktivitäten interpretiert werden, aber bei der genaueren Evaluation wird das korret erfasste Events mit hoher Wahscheinlichkeit der Aktivität zugeordnet.

6 Zusammenfassung und Ausblick

Das Ziel der vorliegen Arbeit war es, einen Algorithmus zu entwerfen und zu implementieren, der unscharfe Zustände innerhalb eines Flows zulässt und mit unscharfen oder unvollständigen Eingabendaten zu einer korrekten Auführung eines Flows führt

In der vorliegenden Arbeit wird ein System umgesetzt, das unscharfe Zustände innerhalb eines Flows zulässt und mit unscharfen oder unvollständigen Eingabedaten zu einer korrekten Ausführung des Flows führt. Dabei müssen zudem fehlerhafte oder fehlende Informationen toleriert werden, um eine weitere Ausführung des Flows zu ermöglichen.

Zunächst wurden die Aufgaben analysiert und die mit dem System gelöst werden sollen und die speziellen Probleme hervorgehoben, die bei der Verarbeitung unscharfer Informationen auftreten können. Auf Basies der Problemanalyse wurden die Eigenschaften des Flowmodells überarbeitet, indem weitere Zustände eingefügt wurden. Die einen bestimmten Zweck für die Robustheit des Systems erfüllen sollen. So erfüllt der *Prepare* Zustand den Zweck, dass einzelne Events, die nicht korrekt erfasst oder interpretiert werden konnten und so der Aktivität fehlen toleriert werden können. Mit Kontepten der Fuzzy Logik wurde die Aufbereitung der Kontextinformationen realisiert, so dass es möglich ist, die Datensätze anhand ihrer Fuzzy Werte zu vergleichen. Zudem lassen sich bei der Zuordnung der Events zu den Aktivitätetn die Eigenschaften der Fuzzy Werte ausnutzen, indem Konflikte über einfache Operationen aufgelöst werden können.

Mit dem EventContainer wurde eine zentrale Komponente eingeführt, die die Verteilung der Events übernimmt und Konflikte auflösen kann. Allgemein wird die Eventverteilung in mehreren Stufen durchgeühführt. Zunächst werden die Aktivitäten über neue Events informiert für deren Empfang sie registriert sind. Sobald für jeden Typ ein Event empfangen wurde, wird die Evaluation der verfügbaren Events angestoßen und geprüft welche der Aktivität zugeordnet werden kann. Erst anschließend werden die Events Angefragt und der Aktivität zugeordnet. Dieses Verfahren bietet die Möglichkeit an mehreren Stellen zu prüfen ob auf dem aktuellen Datenbestand die Interpretation der Daten so gut wie möglich ist.

Für die Evaluation des entwickelten Systems wurden unterschiedliche Parameter definiert. Zum einen wurde die Zuverlässigkeit der Kontexinformationen immer weiter herabgesenkt. Zusätzlich wurden in den Testläufen die Anzahl der nicht erfassten Events erhöht, die Reihenfolge in der Eventversorgung geändert und neue Events mit zufälligen Daten eingefügt, die das fehlerhafte Interpretieren von Umgebunggeräuschen als Aktivität simulieren soll.

Für Eventdaten, bei den die Sicherheit eines Events 40% haben brachte die Simulation ein ordentliche Erfolgsquote mit über 75%. Wurde die Sicherheit allerding unter 40% gesenkt viel dieser Wert auch rapide ab.
Die Simulation hat ergeben, dass wenige vermisste Events gut von dem System toleriert werden können. Überschreitet der Anteil der vermissten Events allerding 10% oder mehr können fast keine Flows mehr erfolgreich abgeschlossen werden.
Den deutlichsten Einfluss hatte die Änderung der Reihenfolge, hier lagen die Ergebnisse deutlich unter den der anderen Testläufe. Das Zusätzliche generieren von Zufälligen Events hatte allerding fast keinen merklichen Einfluss auf die Ausführung der Flows

Die Verwendung der Fuzzy Logik in einer prozess-basierten Anwendung ist ein sehr interessanter Ansatz. Der sich vor allem in der Kombination mit unsicheren Daten auszeichnet. Damit können hier einfach und flexible Interpretationen der Kontexdaten umgesetzt werden. Durch die wenigen und einfachen Regeln lässt sich sehr gut das Verhalten des System bestimmen. Das Problem bei diesem Ansatz ist, das hier optimistisch davon ausgegangen wird, dass der Flow korrekt ausgeführt wurde und dabei überwiegend versucht wird alle Events zu erkennen und richtig verarbeiten zu können. In einem Szenario in dem zusätzlich geprüft werden sollte, ob alle Aktivitäten eines Flows ausgeführt wurde kann der hier gewählte Ansatz zu Problemen führen und sehr dabei sehr viel schlechtere Ergebnisse liefern.

Kontextbasierte Anwendungen sind ein sehr aktuelle Thema und die Integraion von allen möglichen Sensoren wird immer wichtiger. Daher wird mit diesem Ansatz eine praktikable Lösung geboten, auf einer Basis von unterschiedlichen Sensoren durch einfache Regeln eine Navigation entwickelt werden kann, die im Bezug auf unsicher Daten einge gute Robustheit aufweist.

Literaturverzeichnis

[Ada] Adam, Oliver Thomas; O.: A Fuzzy Based Approach to the Improvement of Business Processes (Zitiert auf Seite 15)

[Ada03] (Zitiert auf Seite 15)

In: *Fuzzy Workflows - Enhancing Workflow Management with Vagueness*. 2003

[BBC97] Brown, P. J. ; Bovey, J. D. ; Chen, X.: Context-aware Applications: from the Laboratory to the Marketplace. In: *IEEE Personal Communications* 4 (1997), October, Nr. 5, 58-64 (Zitiert auf Seite 10)

[Bro98] Brown, P.: Triggering information by context. In: *Personal and Ubiquitous Computing* 2 (1998), S. 18–27. – ISSN 1617–4909 (Zitiert auf Seite 10)

[BS03] Buchholz, Thomas ; Schiffers, Michael: Quality of Context: What It Is And Why We Need It. In: *In Proceedings of the 10th Workshop of the OpenView University Association: OVUA'03*, 2003 (Zitiert auf Seite 16)

[DA99] Dey, Anind K. ; Abowd, Gregory D.: Towards a better understanding of context and context-awareness. In: *In HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, Springer-Verlag, 1999, S. 304–307 (Zitiert auf den Seiten 9 und 10)

[DMC+98] Davies, Nigel ; Mitchell, Keith ; Cheverst, Keith ; Blair, Gordon ; Cheverst, Keith ; Blair, Gordon: *Developing A Context Sensitive Tourist Guide*. 1998 (Zitiert auf Seite 10)

[FKS97] Fickas, Stephen ; Kortuem, Gerd ; Segall, Zary: Software Organization for Dynamic and Adaptable Wearable Systems. In: *In Proceedings First International Symposium on Wearable Computers (ISWC'97*, 1997, S. 13–14 (Zitiert auf Seite 10)

[HI04] Henricksen, Karen ; Indulska, Jadwiga: Modelling and Using Imperfect Context Information, 2004, S. 33–37 (Zitiert auf Seite 11)

[HNBr97] Hull, Richard ; Neaves, Philip ; Bedford-roberts, James: Towards Situated Computing. In: *In Proceedings of The First International Symposium on Wearable Computers*, 1997, S. 146–153 (Zitiert auf Seite 10)

[KH05] Krause, Michael ; Hochstatter, Iris: Challenges in Modelling and Using Quality of Context (QoC). In: Magedanz, Thomas (Hrsg.) ; Karmouch, Ahmed (Hrsg.) ; Pierre, Samuel (Hrsg.) ; Venieris, Iakovos (Hrsg.): *Mobility Aware Technologies and Applications* Bd. 3744. Springer Berlin / Heidelberg, 2005, S. 324–333 (Zitiert auf Seite 16)

[Loo06] Loos, Oliver Thomas; Otmar Adam; P.: Using Reference Models for Business Process Improvement: A Fuzzy Paradigm Approach. In: Mayr, W. Abramowicz; H. C. (Hrsg.): *Business Information Systems: 9th International Conference on Business Information Systems (BIS 2006). Köllen Klagenfurt*. Köllen Klagenfurt, Bonn, 6 2006, S. 47–57 (Zitiert auf Seite 15)

[LSI+] Lei, Hui ; Sow, Daby M. ; II, John S. D. ; Banavar, Guruduth ; Ebling, Maria R.: *The Design and Applications of a Context Service* (Zitiert auf Seite 11)

[Pas98] Pascoe, Mr. J.: Adding Generic Contextual Capabilities to Wearable Computers. In: *Proceedings of the 2nd IEEE International Symposium on Wearable Computers*. Washington, DC, USA : IEEE Computer Society, 1998 (ISWC '98). – ISBN 0–8186–9074–7, 92– (Zitiert auf Seite 10)

[Pra] Prabhakar, Sunil: *ORION: Managing Uncertain (Sensor) Data* (Zitiert auf Seite 11)

[Rot08] Rothermel, Kurt: Kontextbezogene Systeme - die Welt im Computer modelliert. In: Rossnagel, Alexander (Hrsg.) ; Sommerlatte, Tom (Hrsg.) ; Winand, Udo (Hrsg.): *Digitale Visionen*. Springer Berlin Heidelberg, 2008. – ISBN 978–3–540–77022–0, S. 31–42 (Zitiert auf Seite 10)

[SAT+99] Schmidt, Albrecht ; Adoo, Kofi A. ; Takaluoma, Antti ; Tuomela, Urop ; Laerhoven, Kristof V. ; Velde, Walter Van D.: Advanced Interaction in Context. In: *In Proceedings of First International Symposium on Handheld and Ubiquitous Computing*, Springer Verlag, 1999, S. 89–101 (Zitiert auf Seite 11)

[See05] Seel, Oliver Thomas; Otmar Adam; C.: A fuzzy based approach to the management of agile processes. In: *Proccedings of the 2nd Workshop on Knowledge Management for Distributed Agile Processes: Models, Techniques, and Infrastructure. 2nd Workshop on Knowledge Management for Distributed Agile Processes: Models, Techniques, and Infrastructure (KMDAP-2005), located at WM 2005, April 10-13, Kaiserslautern, Germany*, DFKI, 2005 (Zitiert auf den Seiten 7 und 15)

[ST94] Schilit, Bill ; Theimer, Marvin: Disseminating Active Map Information to Mobile Hosts. In: *IEEE Network* 8 (1994), S. 22–32 (Zitiert auf Seite 10)

[STL01] Schmidt, Albrecht ; Teco, Albrecht S. ; Laerhoven, Kristof V.: *How to Build Smart Appliances*. 2001 (Zitiert auf Seite 11)

[WHR10] Wolf, Hannes ; Herrmann, Klaus ; Rothermel, Kurt: Robustness in Context-Aware Mobile Computing. In: *IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob'2010)*. Niagara Falls, Canada : IEEE Communications Society, Oktober 2010 (Zitiert auf Seite 8)

[WKL+09] Wieland, Matthias ; Käppeler, Uwe-Philipp ; Levi, Paul ; Leymann, Frank ; Nicklas, Daniela: Towards Integration of Uncertain Sensor Data into Context-aware Workflows. (2009) (Zitiert auf den Seiten 7 und 15)

[WKNL07] Wieland, Matthias ; Kopp, Oliver ; Nicklas, Daniela ; Leymann, Frank: *Towards Context-aware Workflows*. 2007 (Zitiert auf Seite 15)

[Zad65] Zadeh, Lotfali A.: Fuzzy sets. In: *Information and Control* 8 (1965), S. 338–353 (Zitiert auf Seite 12)

[Zad73] Zadeh, Lotfi A.: Outline of a New Approach to the Analysis of Complex Systems and Decision Processes. In: *Systems, Man and Cybernetics, IEEE Transactions on* SMC-3 (1973), Nr. 1, S. 28 –44. – ISSN 0018–9472 (Zitiert auf Seite 12)

Erklärung

Hiermit versichere ich, diese Arbeit selbständig verfasst und nur die angegebenen Quellen benutzt zu haben.

(Jonas Palauro)