

# **Diss**

Stefan Oppl

19. März 2009



# Inhaltsverzeichnis

<b>1 Articulation Work</b>	<b>1</b>
1.1 Begriffsbestimmung . . . . .	1
1.2 Arten von Articulation Work . . . . .	3
1.3 Unterstützung von Articulation Work . . . . .	4
1.4 Fazit . . . . .	4
<b>1 Umsetzung</b>	<b>7</b>
<b>2 Implementierung – Überblick</b>	<b>9</b>
2.1 Grundlegende & verwandte Arbeiten . . . . .	9
2.1.1 Tangible Interfaces . . . . .	10
2.1.2 Historische Entwicklung von Tabletop Interfaces . . . . .	10
2.1.3 Tangible Interfaces zur Modellbildung . . . . .	10
2.1.4 Aktuelle verwandte Ansätze . . . . .	10
<b>3 Input &amp; Interpretation</b>	<b>13</b>
3.1 Möglichkeiten zur Erfassung von Benutzerinteraktion . . . . .	13
3.1.1 In Frage kommende technologische Ansätze . . . . .	14
3.1.2 In Frage kommende Frameworks . . . . .	21
3.1.3 Technologieentscheidung . . . . .	27
3.2 Konzeption und Umsetzung der Hardwarekomponenten . . . . .	32
3.2.1 Überblick . . . . .	32
3.2.2 Tokens & Input-Werkzeuge . . . . .	34
3.2.3 Input auf der Tischoberfläche . . . . .	36
3.2.4 Illumination und Umgebungslichtabhängigkeit . . . . .	37
3.3 Erfassung der Benutzerinteraktion durch Software . . . . .	37
3.4 Interpretation der Rohdaten und Stabilisierung der Erkennungsleistung . . . . .	37
<b>4 Visualisierung und Modellierungsunterstützung</b>	<b>39</b>
4.1 Technologische Grundlage der Visualisierung . . . . .	39
4.1.1 JHotDraw – Überblick . . . . .	39

## Inhaltsverzeichnis

---

4.1.2	Einsatz von JHotDraw . . . . .	39
4.1.3	Projektion von Information auf die Tischoberfläche . . . . .	39
4.2	Umsetzung der Anforderungen zur Modellierungsunterstützung . . . . .	39
4.2.1	Benennung von Blöcken und Verbindungen . . . . .	39
4.2.2	Abstraktion . . . . .	39
4.2.3	Modellierungshistorie . . . . .	39
4.2.4	Wiederherstellungsunterstützung . . . . .	39
<b>5</b>	<b>Persistierung</b>	<b>41</b>
5.1	Möglichkeiten der Persistenzsicherung . . . . .	41
5.2	Topic Maps . . . . .	41
5.3	Abbildung von Modellen auf Topic Maps . . . . .	41
5.4	Technische Umsetzung der Persistierung von Modellen . . . . .	41
<b>6</b>	<b>Untersuchungsdesign</b>	<b>43</b>
6.1	Frage 1 – Unterstützung von Articulation Work . . . . .	43
6.2	Frage 2 – Beitrag und Verwendung der Teilwerkzeuge . . . . .	44
6.3	Untersuchungsablauf . . . . .	45
6.3.1	Phase 1 – Verwendbarkeit . . . . .	45
6.3.2	Phase 2 – Lehr- und Lern-Szenarien . . . . .	45
6.3.3	Phase 3 – Unternehmenseinsatz . . . . .	45
<b>7</b>	<b>Untersuchungsergebnisse</b>	<b>47</b>
7.1	Erhobene Daten . . . . .	47
7.1.1	Phase 1 . . . . .	47
7.1.2	Phase 2 . . . . .	47
7.1.3	Phase 3 . . . . .	47
7.2	Auswertung & Interpretation . . . . .	47
<b>Literaturverzeichnis</b>		<b>49</b>
<b>Index</b>		<b>50</b>

# **Abbildungsverzeichnis**

1.1	Struktur von Arbeitsabläufen . . . . .	2
1.2	Konzeptualisierung von „Arbeit“ nach (Strauss, 1985) und (Fujimura, 1987) . . . . .	3
3.1	ReacTIVision Code . . . . .	26
3.2	Überblick über den Aufbau des Tabletop Interfaces . . . . .	33
3.3	An Tokens angebrachte ReacTIVision-Codes zur Identifikation . . . . .	34
3.4	Arten von Modellierungstokens . . . . .	35
3.5	Rückwand von Container Tokens . . . . .	36
3.6	Geöffnetes Container Token . . . . .	36



# **Tabellenverzeichnis**



# I Articulation Work

In diesem Kapitel wird das Konzept "Articulation Work" dargestellt und in den Kontext von menschlicher Arbeit an sich gestellt. Der zweite Teil des Kapitels widmet sich den Aktivitäten, die »Articulation Work« ausmachen, den Merkmalen, an denen sich gute "Articulation Work" zeigt, sowie den Möglichkeiten der Unterstützung von "Articulation Work" durch organisationale und technische Maßnahmen.

## I.I Begriffsbestimmung

Das Konzept der "Articulation Work" wurde als Erklärungsmodell für einen bestimmten Typus von menschlicher Arbeit Mitte der 1980er Jahre von Strauss (1985) im Kontext von Fallstudien aus der Krankenhaus-Organisation eingeführt. "Articulation Work" ist dabei jener Anteil an menschlicher Arbeit, der der Abstimmung mit anderen Individuen dient. Diese Abstimmung ist notwendig um das eigentliche Arbeitsziel erreichen zu können. Arbeit wird als inhärent kooperative Prozess gesehen, der immer auf Interaktion mit anderen Menschen basiert bzw. diese bedingt (Strauss formuliert diese Annahme in Bezugnahme auf Hughes (1971) prägnant mit der Aussage "*work rests ultimately on interaction*"). Diese Annahme erscheint insofern als zulässig, als dass selbst Arbeitsabläufe, die selbst keine Kooperation mit anderen Menschen bedingen, zumindest auf den Ergebnissen anderer Arbeitsabläufe aufbauen oder als Grundlage weiterer Arbeitsabläufe dienen. Interaktion tritt also in jedem Arbeitsprozess zumindest zu Beginn und am Ende in unmittelbarer oder mittelbarer<sup>1</sup> Form auf.

**Abbildung, in der kooperative Arbeitsprozesse und solche mit mittelbarer und unmittelbarer Interaktion zu Beginn oder am Ende dargestellt werden**

Jener Teil von Arbeit, der der eigentlichen Zielerreichung dient, wird im hier vorgestellten Erklärungsmodell als "Production Work" bezeichnet (Fujimura, 1987). "Production Work" ist komplementär zu "Articulation Work" zu sehen und umfasst

---

<sup>1</sup>Unter "mittelbar" ist hier Interaktion zu verstehen, die nicht im direkten Kontakt zwischen Individuen abläuft sondern lediglich indirekt durch die Ergebnisse eines Arbeitsprozesses (Materialien, Dokumente, ...) vermittelt wird.

## 1 Articulation Work

---

alle Aktivitäten, die der "Wertschöpfung" im wörtlichen Sinn, also der Schaffung jener Werte (oder Ergebnisse), die durch den Arbeitsablauf erreicht werden sollten. Wenn hier von Arbeit bzw. Arbeitsabläufen die Rede ist, so ist darunter eine Kette von Aktivitäten zu verstehen, die der Erreichung einer vorab gewählten Ziels dient<sup>2</sup>.

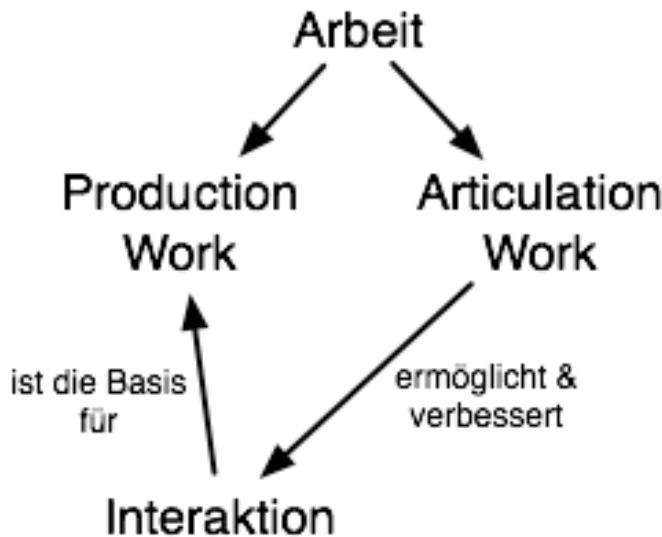


Abbildung 1.1: Struktur von Arbeitsabläufen

Teile eines Arbeitsablaufs dienen also der Zielerreichung an sich ("Production Work"). Andere Teile dienen der Abstimmung zwischen den involvierten Akteuren, um ein gemeinsames Verständnis über die jeweiligen Schnittstellen – also die Beührungspunkte zwischen den Tätigkeiten – zu entwickeln. Diese "Koordination" ist kritisch für den Erfolg von kooperativer Arbeit (Strauss, 1993) und wird als "Articulation Work" bezeichnet.<sup>3</sup> "Articulation Work" ist damit ein Enabler für funktionierende Kommunikation und Koordination im eigentlichen Arbeitsprozess<sup>4</sup>.

Der Begriff "Articulation Work" ist im Englischen zweideutig und von Strauss bewusst so gewählt. Einerseits wird damit ausgedrückt, dass *Arbeit* ("Work") arti-

<sup>2</sup>siehe dazu etwa die Definition von Arbeit durch Semmer und Udris (2004): "Arbeit ist zielgerichtete menschliche Tätigkeit zum Zwecke der Transformation und Aneignung der Umwelt aufgrund selbst- oder fremddefinierter Aufgaben, mit gesellschaftlicher, materieller oder ideeller Bewertung zur Realisierung oder Weiterentwicklung individueller oder kollektiver Bedürfnisse, Ansprüche und Kompetenzen."

<sup>3</sup>"Without an understanding of articulation, the gap between requirements and the actual work process in the office will remain inaccessible to analysis. That is, it will be possible to describe tasks in an idealized form but not to describe actual situations."(Gerson and Star, 1986)

<sup>4</sup>"Reconciling incommensurate assumptions and procedures in the absence of enforceable standards is the essence of articulation. Articulation consists of all the tasks involved in assembling, scheduling, monitoring, and coordinating all of the steps necessary to complete a production task."(Gerson and Star, 1986)

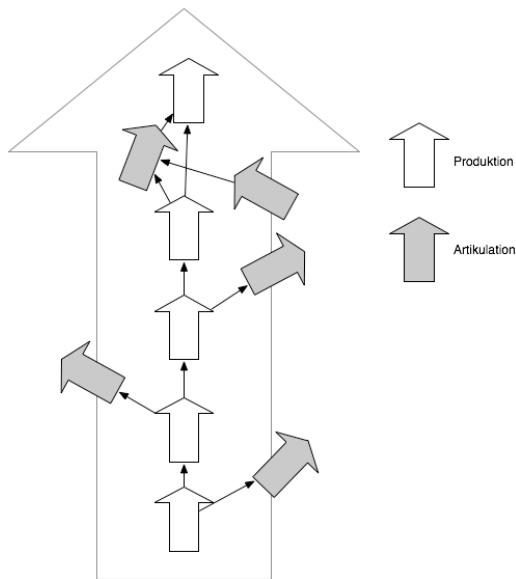


Abbildung 1.2: Konzeptualisierung von „Arbeit“ nach (Strauss, 1985) und (Fujimura, 1987)

kuliert wird, andererseits zeigt der Begriff, dass die *Artikulation* selbst ebenfalls Arbeit ist (also Zeit und Ressourcen in Anspruch nimmt) und auch also solche wertgeschätzt werden muss (Fujimura, 1987). „Articulation Work“ ist kein klar abgegrenztes und strukturiertes Konzept – sie tritt je nach Arbeitssituation in unterschiedlichen Spielarten auf. Die Unterscheidung dieser Arten von „Articulation Work“ ist für die Unterstützung derselben relevant und wird daher im folgenden Abschnitt genauer betrachtet.

## I.2 Arten von Articulation Work

Strauss argumentiert, dass Artikulation immer passieren muss (und passiert), wo Menschen zusammenarbeiten, um zu vermeiden, dass unbekannte Aspekte Probleme bei der Durchführung der Arbeit verursachen (Strauss, 1988). „Articulation Work“ ist kein revolutionäres Konzept, sondern fasst Tätigkeiten unter einem Begriff zusammen, die seit jeher Teil jeder Zusammenarbeit zwischen Menschen sind. Grundsätzlich geht Strauss davon aus, dass Artikulation immer abläuft, egal wie einfach oder kompliziert, wie eingespielt oder neuartig eine (Zusammen-)Arbeit ist (Strauss, 1988). Sehr wohl existieren jedoch Unterschiede in der Qualität der Arbeit, die sich auf die Form der Artikulation auswirken, die zu deren Abstimmung notwendig ist: *“A useful fundamental distinction between classes of interaction is bet-*

*ween the routine and the problematic. Problematic interactions involve 'thought', or when more than one interactant is involved then also 'discussion'.*" (Strauss, 1993). Dieses Zitat zeigt im Übrigen auch, dass "Interaction" im Sinne von Strauss nicht unbedingt ein kollektives Phänomen ist, sondern auch individuell (im Bezug auf die (unbelebte) Umgebung) auftreten kann.

Je komplexer ("problematic") eine Interaktion ist, desto notwendiger wird laut Strauss eine explizite Beschäftigung mit dem Vorgang der Artikulation. Bei einfachen, eingespielten ("routine") Interaktionen bleibt die Artikulation zumeist implizit, verborgen und informell (Hampson and Junor, 2005) (entsprechend der "Sozialisation" in Nonaka & Takeuchis SECI-Zyklus (Nonaka and Takeuchi, 1995)). Ein grundlegendes Problem, dass Artikulation für jeden noch so einfach erscheinend Arbeitsvorgang potentiell relevant macht, spricht Strauss mit den Worten von Hughes unmittelbar nach der Definition von "problematic interaction" an: "[O]ne man's routine of work is made up of the emergencies of other people" (Hughes, 1971) zitiert nach (Strauss, 1993).

"Articulation Work" tritt also in zwei Qualitäten auf. Ist der Bedarf zur Abstimmung bekannt und werden Tätigkeiten zur Abdeckung dieses Bedarf bewusst durchgeführt, so spricht man von *expliziter* "Articulation Work" (Strauss, 1988)(Fjuk et al., 1997). Die Abstimmung von Tätigkeiten, die ständig während der Zusammenarbeit unbewusst ausgeführt wird, bezeichnet man als *implizite* "Articulation Work". Letztgenannte Art ist es auch, die von den Arbeitenden "automatisch" zur Anwendung gebracht wird, sobald Änderungen in der Arbeitsumgebung oder Probleme auftreten (Strauss, 1988). Implizite "Articulation Work" stößt aber an ihre Grenzen, wenn die Arbeitssituation als "problematisch" (Strauss, 1988) oder "komplex" (Schmidt, 1990, S. 23f) wahrgenommen wird. Es wird dann notwendig, dezidierte Abstimmungs-Aktivitäten anzustossen, also explizite "Articulation Work" durchzuführen.

### I.3 Unterstützung von Articulation Work

### I.4 Fazit

**hier muss eine zusammenfassende Tabelle der in der Literatur verfügbaren Information rein**

Die Zielsetzung von „Articulation Work“ formulieren die Proponenten des Ansatzes - allen voran Strauss - klar aus. Offen bleiben jedoch bei allen Autoren direkten Aussagen zum eigentlichen Gegenstand von „Articulation Work“ – also Allem was von den beteiligten Individuen zu artikulieren ist – und den notwendigen Leistungen

der Individuen im Prozess der Artikulation. Aussagen zu diesen Aspekten sind aber für die Entwicklung von Ansätzen zur Unterstützung von expliziter „Articulation Work“ notwendig.

Strauss ist sich dieser Auslassung bewusst<sup>5</sup>, und beschäftigt sich in späteren Arbeiten (Strauss, 1993) auch mit jenen kognitiven Vorgängen, die von ihm als "thought processes" oder "mental activities" bezeichnet werden und die untrennbar mit jeder Art von Tätigkeit und Interaktion verbunden sind<sup>6</sup> und diese beeinflussen<sup>7</sup>.

Im Kontext der Abstimmung von Tätigkeiten kommt den "thought processes" der Individuen große Bedeutung zu, da sie den sichtbaren individuellen Handlungen zugrunde liegen bzw. diese beeinflussen. "Articulation Work" wirkt sich also auf die "thought processes" der beteiligten Individuen aus. "Thought processes" umfassen "*images, imaginations, projections of scenes, [...] flashes of insight, rehearsals of action, construction and reconstruction of scenarios, the spouting up of metaphors or comparisons, the reworking and reevaluating of past scenes and one's actions within them, and so on and on*" (Strauss, 1993, S. 130) - also im Wesentlichen alle kognitiven Vorgänge, die unmittelbar oder mittelbar im Zusammenhang mit den sichtbaren Arbeitsspekten, insbesondere den Tätigkeiten zur Zielerreichung und der wahrgenommenen Arbeitsumgebung, stehen. Strauss interessiert sich allerdings ausschließlich für die dynamischen Aspekte der Interaktion zwischen Individuen, nicht aber für die Ausgangspunkte und Ergebnisse der zugrunde liegenden "thought processes".<sup>8</sup> Wie bereits oben erwähnt sind aber die Repräsentationen, auf den „thought processes“ beruhen und operieren, für die Unterstützung von "Articulation Work" von Interesse. Die kognitions-wissenschaftlichen Ansätze zu Schemata (Rumelhart and Norman (1978) Hanke (2006, vgl. nach )) und mentalen Modellen (Seel (1991, vgl. )) sind ein Erklärungsansatz für diese Lücke.

---

<sup>5</sup>"[...] many social scientist pay almost no attention to interior activity: ignoring it, taking it for granted, but leaving it unexamined, or giving it the kind of abstract but not very detailed analysis [...]"(Strauss, 1993, S. 131)

<sup>6</sup>"These [thought processes] accompany visible action, as well as precede and follow in conditional and consequential modes"(Strauss, 1993, S. 146)

<sup>7</sup>"Even well-grooved, routine action and interaction may be accompanied by thought [...] directly relevant to the work at hand. As I vacuum the house, barely noticing my movements, still I give myself commands [...]"(Strauss, 1993, S. 132)

<sup>8</sup>"I use the gerund 'ing' after 'symbol' [bei der Beschreibung von 'symbolizing', Anm.] to signify that my principal interest is, again, in interaction rather than its products, for symbols are precipitates of interaction"(Strauss, 1993, S. 149)



# **Teil I**

# **Umsetzung**



## **2 Implementierung – Überblick**

Wie im Kapitel "Design" gefordert, wurde zur Umsetzung des Werkzeugs ein "Tangible Tabletop Interface" verwendet. Tabletop Interface zeichnen sich im Generellen dadurch aus, dass im Gegensatz zu handelsüblichen Rechnern nicht nur die Software sondern auch die Hardware applikationsspezifisch ist und nicht generisch eingesetzt werden kann. Die Hardware bildet dabei einen Teil oder die gesamte Benutzungsschnittstelle ab. Im speziellen Fall eines "Tangible Tabletop Interfaces" basiert der Benutzerinteraktion auf der Verwendung physischer Bausteine ("Tokens"), die auf der physischen Oberfläche des Interfaces manipuliert werden. Dieses Paradigma wird ergänzt von Tabletop Interfaces, die die Benutzerinteraktion ausschließlich auf Gesten bzw. Berührungen der Oberfläche abbilden (horizontal verbaute "Touch-" bzw. "Multi-Touch-Displays").

### **2.1 Grundlegende & verwandte Arbeiten**

REFs!!! Die Entwicklung von Tabletop Interfaces begann Mitte der 1990er-Jahren mit den Arbeiten von Ishii & Ullmer. Auch die erste Anwendung, die sich mit Modellierungs-Ansätzen mit Hilfe von Tabletop Interfaces konzentriert, stammt aus dieser Zeit. Mit dem fortschreiten der technologischen Entwicklung ist heute ein Status erreicht, in dem mit Hilfe generischer Identifikations-Frameworks schnell und ohne großen Aufwand Applikationen mit "tangiblen" Inputkanälen erstellt werden können. Zur Zeit noch im Prototypenstatus befinden sich Ansätze, die sich mit generischen Möglichkeiten des tangiblen Informationsoutputs beschäftigt. Der Rückkanal vom Rechner zum Benutzer wird heute zumeist mit der Projektion von Inhalten auf die Arbeitsoberfläche umgesetzt.

In den folgenden Abschnitten wird die historische Entwicklung von Tabletop Interfaces sowie der aktuelle Stand der Entwicklung im Anwendungsbereich dieser Arbeit betrachtet. Es werden dabei die grundlegenden Konzepte und Eigenschaften der jeweiligen Arbeiten betrachtet und das Potential hinsichtlich der Umsetzung von in Kapitel XY identifizierten Anforderungen an das hier entwickelte Werkzeug betrachtet.

## 2.1.1 Tangible Interfaces

Der Begriff der Tangible bzw. Graspable Interfaces – also der "berührbaren" oder "begreifbaren" Benutzungsschnittstellen — stammt aus der Mitte der neunziger Jahre des zwanzigsten Jahrhunderts. Fitzmaurice et al. (1995) werden im Allgemeinen als die ersten betrachtet, die den Begriff des "Graspable User Interfaces" prägen und damit die Manipulierbarkeit digitaler Information durch physische Mittel beschreiben. Fitzmaurice (1996) präzisiert später den Begriff durch die Abgrenzung zwischen (herkömmlichen, maus-, tastatur- und bildschirmbasierenden) zeitlich gemultiplexten Schnittstellen, bei denen der Informationsaustausch zwischen Benutzer und System über einen Kanal zeitlich hintereinander erfolgt und den (neuartigen, berührbaren) räumlich gemultiplexten Schnittstellen, bei denen mehrere Kanäle gleichzeitig zur Interaktion zwischen Benutzer und System verwendet werden können.

Der Begriff des "Tangible User Interfaces" wurde kurz danach bzw. parallel dazu von Ishii and Ullmer (1997) eingeführt. Ishii and Ullmer verfolgen dabei bei der Definition den umgekehrten Weg und sprechen von einer "Augmentation der realen Welt durch eine Kopplung von digitaler Information and physische Objekte".

## 2.1.2 Historische Entwicklung von Tabletop Interfaces

**Sensetable** Der Sensetable (Patten et al., 2001)

**BUILD-IT** (Fjeld, 2001)

## 2.1.3 Tangible Interfaces zur Modellbildung

## 2.1.4 Aktuelle verwandte Ansätze

- Historische Entwicklung von Tabletop Interfaces
  - Sensetable
  - Morten Fjeld
  - ReacTable
  - Eva Hornecker
- Historische Entwicklung von Tangible Interfaces zur Modellbildung

---

*"augment the real physical world by coupling digital information to everyday physical objects and environments"*(Ishii and Ullmer, 1997)

- Sensetable Modeling Application
- Designer's Outpost (Klemmer)
- Aktuelle verwandte Ansätze
  - Antle (TEI Mail-Pointer)
  - Sun (TEI Demo)



# **3 Input & Interpretation**

In diesem Kapitel wird jener Teil des Werkzeuges beschrieben, in dem die Interaktion der Benutzer mit dem Werkzeug erfasst und interpretiert wird. Der erste Abschnitt behandelt grundlegende Möglichkeiten zur Erfassung der Benutzerinteraktion auf tisch-basierten Benutzungsschnittstellen und endet mit der Identifikation der im konkreten Anwendungsfalls geeigneten Technologie. Diese wird durch die Beschreibung von dafür verfügbaren Frameworks konkretisiert, was letztendlich in der Entscheidung für ein konkretes Produkt mündet.

Basierend auf dieser Entscheidung wird in den folgenden beiden Abschnitten auf das Design der Hard- und Softwarekomponenten eingegangen, die unmittelbar der Eingabe von Information durch Benutzer dienen. Der Ausgabeaspekt wird hier bewusst ausgeklammert und im nächsten Kapitel beschrieben. Dieses Kapitel endet mit einer Beschreibung der Interpretationsroutinen, die aus den durch das Framework gelieferten Rohdaten höherwertige, anwendungsspezifische Information extrahieren und diese den nachgeordneten Software-Modulen zur Verfügung stellen.

## **3.1 Möglichkeiten zur Erfassung von Benutzerinteraktion**

Im Gegensatz zu Systemen mit dezidierten Eingabegeräten (wie Tastatur oder Maus) ist die Informationseingabe bei Tangible Interfaces unmittelbar an physische Tokens gebunden, die unabhängig voneinander und gegebenenfalls auch simultan manipuliert werden können. Diese Manipulation wird von einer vorhandenen Infrastruktur erfasst und im Sinne von Eingabedaten interpretiert. Der wesentliche Unterschied zu dezidierten Eingabegeräten besteht darin, dass die Manipulation des physischen Artefakts selbst für den Benutzer bedeutungstragend ist und nicht nur dem Zweck einer Zustandsänderung des digitalen Informationsraums dient. Dies impliziert, dass der Zustand der verwendeten Tokens bzw. der aktuelle Wert deren relevanten Parameter (z.B. Position, Rotation, Form, ...) erfasst werden kann, ohne die Bedeutung der Tokens noch deren Manipulierbarkeit in der realen Welt zu beeinflussen. Je nach Anwendungsfall kommen dafür mehrere unterschiedliche technologische Ansätze in Frage. Die Beurteilungskriterien die dabei zu berücksichtigen

sind, liegen nicht nur in den zu erhebenden Parametern begründet sondern umfassen auch die notwendige Erfassungsrate des Zustandes der Tokens sowie die Anzahl der simultan zu erfassenden Tokens bzw. Eigenschaften.

Im konkreten System muss - wie in Kapitel XY beschrieben - die planare Position von mehreren Tokens in Echtzeit (d.h. mehrmals pro Sekunde mit für den Benutzer nicht wahrnehmbaren Verzögerungen) erfasst werden. Neben der Position ist noch die Rotation eines Tokens als Raumparameter von Interesse. Bezuglich des Zustands eines Tokens muss erfasst werden können, ob es geöffnet oder geschlossen ist und ob es eingebettete Objekte enthält oder nicht. Mit diesen Anforderungen wird in den folgenden Unterabschnitten ein technologischer Ansatz zur Umsetzung der Interaktionserkennung ausgewählt.

#### **3.1.1 In Frage kommende technologische Ansätze**

Bei der Auswahl möglicher technologischer Ansätze zur Erfassung der Benutzerinteraktion müssen die zu erfassenden Parameter unterschiedlich behandelt werden. Konkret werden hier Ansätze zur Erfassung der Raumparameter (Position, Rotation) und Ansätze die Zustandsänderungen des Tokens erfassbar machen unterscheiden.

##### **Raumparameter**

Zur Erfassung von Raumparametern von Tokens bieten sich mehrere technologische Ansätze an. In Frage kommen für das konkrete - tisch-basierte System - nur Technologien, die eine Erfassung dieser Parameter mit einer Genauigkeit im Zentimeter bis Millimeter-Bereich ermöglichen, da eine niedrigere Raumauflösung zu zu großen Ungenauigkeiten in der Positionsbestimmung führen würden, die einen Einsatz für das hier vorgestellte Werkzeug nicht erlauben würden. Im Folgenden werden die in Frage kommenden Technologien in ihren Grundzügen beschrieben und hinsichtlich ihrer Eignung für das konkrete System bewertet.

**Optisch** Optische Positionsbestimmung erfolgt immer mit Hilfe von Kamera-Systemen und Methoden der digitalen Bildverarbeitung. Die Kamera erfasst dabei die zu identifizierenden Tokens, dass resultierende Bild wird mit in Software umgesetzten Algorithmen ausgewertet, wodurch zumindest Identität und Position, zumeist aber auch weitere Raumparameter (wie Rotation) aller im Kamerabild befindlichen Tokens ermittelt werden können. Die Erkennung ist dabei auf den Erfassungsbereich der Kamera beschränkt. Neben diesem Einflussfaktor bestimmen zudem die Auflösung der Kamera sowie die Größe der Tokens die letztendlich er-

fassbare Fläche. Optische Systeme sind generell bei schlechten oder wechselnden Lichtverhältnissen eher fehleranfällig und nicht robust gegen Verdeckungen von Tokens (etwa durch Gliedmaßen oder andere Tokens).

Hinsichtlich des Identifikationsansatzes können zwei Arten von Systemen unterschieden werden. *Codebasierte* Systeme verwenden zur Identifikation eines Tokens einen von der Kamera erfassbaren Code (etwa einen "Barcode"), der eindeutig einem Token zugeordnet werden kann. *Featurebasierte* Systeme identifizieren ein Token aufgrund seiner äußereren Eigenschaften, zumeist über dessen Form (Schattenriss). Letztere bieten den Vorteil, dass ein Token nicht durch das Anbringen eines zusätzlichen Codes optisch verändert werden muss. Der größte Nachteil besteht in der Eigenschaft, dass nur Token mit unterschiedlichen Formen eindeutig identifiziert werden können. Die eindeutige Identifikation von mehreren Tokens einer Bauart ist bei featurebasierten Systemen nicht möglich. Codebasierte Systeme verwenden zumeist nicht herkömmliche Barcodes sondern robustere Systeme, bei denen eine Erkennung auch unter widrigen Beleuchtungsbedingungen oder niedrigen Bildauflösungen möglich ist und die zum Teil auch die Extraktion zusätzliche Information über weitere Raumparameter (wie Rotation, teilweise auch Parameter der dritten Dimension wie Neigung oder Entfernung) ermöglichen.

Codebasierte Systeme können hinsichtlich der Art der Codierung der Identitätsinformation wiederum in zwei Klassen unterschieden werden. Eine Gruppe von Ansätzen integriert die eigentliche Nutzinformation, also im Wesentlichen die tokenspezifische Identifikationsnummer, direkt in den Code und ermöglicht so ein direktes Auslesen der Information (z.B. bei QRCode (REF)). Die zweite Gruppe verwendet eine indirekte Zuordnung zwischen Token-ID und Code. Bei derartigen Ansätzen muss die Identität eines Tokens in einem Zwischenschritt über eine Mapping-Tabelle abgebildet werden, im Gegenzug ist die Ausgestaltung des Codes flexibler, im Allgemeinen kann dabei eine höhere Robustheit bei der Erkennung erreicht werden (z.B. ARToolkit (REF)).

**Kapazitiv** Kapazitive Ansätze basieren auf der Änderung der Kapazität von Leiterbahnen, die durch deren Berührung mit leitfähigem Material verursacht wird. Ursprünglich wurde die Technologie zur Umsetzung von berührungssensitiven Oberflächen entwickelt, kann jedoch auch zum Tracking von Tokens verwendet werden. Im Gegensatz zu druckempfindlichen Oberflächen (klassischen "Touchscreens") ist keine Druckausübung zur Erkennung notwendig, es können außerdem auch mehrere Tokens (bzw. Finger) gleichzeitig erkannt werden.

Technologisch bedingt müssen bei kapazitiven Ansätzen alle zu identifizierenden Objekte die Oberfläche des Systems berühren. In dieser Oberfläche ist ein Metallgitter eingebettet, zwischen dessen Adern eine elektrische Kapazität gemessen wer-

den kann. Diese Kapazität verändert sich, sobald diese Adern berührt werden (wobei die Token in einen entsprechend geeigneten Material ausgeführt sein müssen). Durch die lokale Änderung der Kapazität kann die Position einer Berührung festgestellt werden. Die Genauigkeit ist dabei durch die Rasterweite des Metallgitters eingeschränkt. Der größte Nachteil eines kapazitiven Ansatzes ist in diesem Kontext aber, dass die Identität eines Tokens nicht direkt festgestellt werden kann (die Kapazitätsänderung ist für alle Token identisch). Zudem ist die Extraktion weiterer Raumparameter (wie Rotation) nicht bzw. nur mit zusätzlichen Aufwand möglich. Die Vorteile von kapazitiven Systemen liegen in der hohen Robustheit der Erkennung auch bei widrigen Umgebungseinflüssen (Lichtverhältnisse, Schmutz) sowie der prinzipiell beliebig großen und beliebig geformten Oberfläche, die zur Erkennung verwendet werden kann.

Kapazitive Systeme eignen sich also zur Positionsbestimmung, nicht aber zur Identifikation von Tokens. Dies macht sie für den konkreten Anwendungsfall nur in Kombination mit einer anderen Technologie geeignet.

**Elektromagnetisch** Die Ausstattung von Tokens mit elektromagnetisch erfassbaren Einheiten (z.B. RFID-Chips) ermöglicht ebenfalls die Erfassung von Raumparametern. Vorrangig eignet sich diese Technologie jedoch zur Identifikation von Tokens, die Positionsbestimmung kann nur mit erheblichem technischen Aufwand durchgeführt werden.

RFID-Chips (als Beispiel für einen elektromagnetischen Ansatz) sind passive Bauteile, die bei Energieversorgung durch ein elektrisches Feld aktiv werden und ihrerseits eine eindeutige Identifikationsnummer senden (im einfachsten Fall, komplexere Varianten sind möglich, werden hier aber nicht betrachtet). Historisch stammt die Technologie aus der Logistik und Warenwirtschaft und dient der Identifikation von Gütern und nicht der exakten Positionsbestimmung. Diese ist somit auch nur mittels erweiterter Infrastruktur möglich. Zum Auslesen eines RFID-Chips wird ein Lesegerät mit Antenne benötigt. Aus der Feldstärke, mit der die Antenne die Antwort des Chips empfängt, kann auf die Entfernung des Chips von der Antenne geschlossen werden. Durch Kreuzpeilung mit mindestens zwei Antennen, deren Position bekannt ist, kann somit auf die ungefähre Position des Chips (und damit des Tokens, in das dieser eingebaut ist) geschlossen werden. Durch den Einsatz von "Antennenarrays" (matrixförmig angeordneten Antennen) mit geringer Reichweite ist so eine verhältnismäßig exakte (Größenordnung einige cm) Positionsbestimmung möglich. Die Feststellung der Ausrichtung eines Tokens (Rotation) ist auf diesem Wege allerdings nicht möglich. Die Identifikation eines Tokens ist jedoch unabhängig von Sichtkontakt und unmittelbarer Berührung und somit äußerst robust gegen Umgebungseinflüsse.

Elektromagnetische Systeme eignen sich wegen des hohen technischen Aufwandes bei gleichzeitig beschränkter Genauigkeit nur bedingt zur Feststellung von Raumparametern. Durch die Ausrichtung auf Extraktion der Identitätsinformation ist der Ansatz jedoch gut zur Kombination mit anderen Technologien wie kapazitiven Ansätzen geeignet, die ihre Stärken in der Bestimmung der Raumparameter haben.

**Akustisch** Akustische Ansätze zur Positionsbestimmung basieren im Generellen auf der Laufzeitmessung von Ultraschallwellen im Raum. Mit entsprechender Infrastruktur ist damit in einem begrenzten Bereich eine hochexakte Feststellung der Raumparameter in drei Dimensionen (Genauigkeit im mm-Bereich) sowie die Identifikation von Tokens möglich.

Ultraschallbasierte Techniken zur Positionsbestimmung basieren auf dem Einsatz von Bakensendern an bekannten Positionen. Diese Sender werden zumeist an der Zimmerdecke montiert und senden periodisch einen Ultraschallimpuls aus. Dieser Impuls wird von den Tokens (die in diesem Fall aktive Bauteile mit Stromversorgung sind) empfangen, die daraufhin einen sie identifizierenden Impuls zurücksenden. Aus der Laufzeit zwischen Absetzen des Sendeimpuls und Empfangen des Antwortimpulses bei verschiedenen Baken lässt sich so die Position des Tokens im Raum feststellen. Problematisch ist hierbei jedoch die durch den auf sequentieller Zeitmessung basierenden Ansatz beschränkte Anzahl von verfolgbaren Tokens, wenn Echtzeit-Ansprüche gestellt werden. Zudem ist der Ansatz nicht robust gegen (akustisch) verdeckte Tokens. Eine Anfälligkeit gegenüber anderen Störeinflüssen besteht nicht.

Für die Feststellung von Raumparametern sind ultraschall-basierte Systeme generell ausgezeichnet geeignet. Auch die Identifikation von Tokens ist prinzipiell möglich. Bei der Bewertung hinsichtlich des Einsatzes für tisch-basierte Systeme ist jedoch zu bedenken, dass eine drei-dimensionale Positionierung nicht zu den allgemeinen Anforderungen zählt und nur in speziellen Anwendungsfällen sinnvoll sein kann. Zudem kann die Notwendigkeit von stromversorgten Tokens einen Nachteil bzw. ein Hindernis beim Einsatz darstellen.

**Bewertung** Im konkreten Anwendungsfällen ist die Feststellung der Identität sowie der planaren Position und Rotation von mehreren Tokens in hoher Genauigkeit sowie in Echtzeit gefordert. Aus oben genannten Gründen sind kapazitive und elektromagnetische Systeme im Einzeleinsatz nur bedingt geeignet. Akustische Systeme erscheinen für den Anwendungsfällen als zu aufwändig und unflexibel und stoßen außerdem bei der Anzahl der simultan zu verfolgenden Tokens an ihre Grenzen.

Die Kombination von kapazitiven und elektromagnetischen Systemen ist grundsätzlich eine Möglichkeit, die in Betracht gezogen werden könnte. Auch optische

### 3 Input & Interpretation

Systeme genügen den Anforderungen und kommen damit in Frage. Der kombinierte Ansatz ist im Vergleich mit optischen Systemen als robuster gegen Störeinflüsse aus der Umgebung zu betrachten. Für optische Systeme sprechen hingegen die weitaus geringeren Aufwände für Infrastruktur und Tokens sowohl bei Anschaffung als auch bei Wartung und Betrieb. Durch die geringere Komplexität des Systems sind auch weniger potentielle Fehlerquellen vorhanden, was bei der Erstellung des Werkzeug-Prototypen hilfreich ist. Aufgrund dieser Aspekte und einer vergleichbaren zur erwartenden Erkennungsleistung wurde für die hier vorgestellten Anwendungsfalls die Entscheidung getroffen, ein optisches System zur Bestimmung der Positionsparameter sowie der Identität der Tokens einzusetzen.

#### **Tokenzustand**

Hinsichtlich des Tokenzustands sind im Kontext des hier vorgestellten Anwendungsfalls Informationen zu erheben, die den Inhalt des Tokens betreffen. Wie in Kapitel XY beschrieben sind die Modellierungs-Tokens als Container ausgeführt, die geöffnet und geschlossen werden können und in die kleiner Tokens als Träger von Zusatzinformation hineingelegt werden können. Die Auswahl eines Ansatzes, der die Identifikation des Öffnungs-Zustandes eines Tokens sowie dessen Inhalt erlaubt, ist Gegenstand dieses Abschnitts. Dazu wird grundlegend zwischen dem Einsatz von passiven Tokens und aktiven Tokens unterschieden. Passive Tokens besitzen keine zusätzliche Elektronik, die geforderten Informationen können lediglich durch die bereits vorhandene (optische) Infrastruktur festgestellt werden. Aktive Tokens werden hingegen mit zusätzlicher Elektronik zur Zustandsbestimmung ausgestattet, was allerdings eine Energieversorgung jedes Tokens bedingt.

**Passive Token** Bei passiven Tokens muss sichergestellt werden, dass die bereits vorhandene Infrastruktur die Zustandsänderungen eines Tokens erfassen kann. Da die vorhandene Infrastruktur auf optischen Technologien basiert, müssen sich alle Zustandsänderungen im äußeren - durch die Kamera erfassbaren - Erscheinungsbild eines Tokens wieder spiegeln.

Der Öffnungszustand eines Tokens kann durch Kameras einfach erfasst werden, wenn sich - je nach eingesetzter Technologie - durch das Öffnen der Umriss des Tokens verändert oder ein weiterer Code sichtbar wird bzw. der bestehende Code modifiziert wird. Diese Anforderung kann also durch passive Tokens erfüllt werden.

Zur Erfassung des Inhalts eines Container-Tokens sind zwei Ansätze denkbar. Einerseits kann der Inhalt eines Tokens zu einem bestimmten Zeitpunkt erfasst werden, andererseits ist auch eine Erfassung der Änderung des Tokeninhalts möglich (Erfassung des Vorgangs von Hineinlegen und Herausnehmen). Diese beiden Mög-

lichkeiten sind hinsichtlich der Umsetzbarkeit mit passiven Token unterschiedlich zu beurteilen. Eine Erfassung des aktuellen Tokeninhalts ist mit optischen Systemen nur schwer möglich. Die einzige sich bietende Möglichkeit ist die von transparenten Teilbereichen der Außenfläche eines Tokens. Damit ist es grundsätzlich möglich, den Inhalt eines Tokens mit einer externen Kamera zu erfassen, sowohl bei feature- als auch code-basierten Ansätzen sind jedoch Verdeckungen, Verzerrungen oder zu geringe Kameraauflösung potentiell problematisch und lassen diesen Ansatz für den praktischen Einsatz als ungeeignet erscheinen.

Die Erfassung der Änderung des Tokeninhalts lässt sich mit optischen Systemen einfach implementieren. So kann der Vorgang des Hineinlegens als auch des Herausnehmens von einer Kamera erfasst werden. Die größte Herausforderung hierbei ist die Identifikation des Tokens, das eingebettet wird. Hier kann es wiederum durch Verdeckungen zu Erkennungsschwierigkeiten führen, was in diesem Fall einen permanent fehlerhaften Modellzustand zur Folge hat, der sich im Falle wiederholter Fehlerkennungen sogar inkrementell verschlimmern kann. Diesem Umstand kann lediglich durch eine explizite Aktion des Benutzers Rechnung getragen werden, der das betreffende einzubettende Token ins Sichtfeld der Kamera halten muss, bis das System Feedback über eine erfolgreiche Erkennung gibt. Diese Lösung erscheint allerdings hinsichtlich der Anforderung, die Technologie für den Benutzer vollkommen in den Hintergrund treten zu lassen, als eher suboptimal.

**Aktive Token** Aktive Tokens beinhalten zusätzliche Sensorik, die die Erfassung des Tokenzustands ermöglicht. Derartige Tokens benötigen allerdings eine Energieversorgung und müssen über eine Möglichkeit zur Datenübertragung verfügen, um den Tokenzustand an das System zu übermitteln. Weiters ist im Allgemeinen eine Steuereinheit notwendig, um die Sensoren zu kontrollieren, die Daten zu aggregieren und letztendlich zu übertagen.

Im konkreten Fall einer optisch arbeitenden Infrastruktur bietet sich eine (ggf. aufladbare) Batterie als Energiequelle an, um im Kamerabild Verdeckungen durch ansonsten eventuell zu verwendende Kabel zu vermeiden. Eine Stromversorgung über die Oberfläche (wie z.B. im Smart PINS (Gellersen REF) Ansatz vergestellt) scheidet hier aus, da die Blöcke dann mit Krafteinsatz auf die Oberfläche gesetzt werden müssten und nicht verschoben werden können.

Als Steuerungseinheit bietet sich neben selbst auf der Basis von Mikrocontrollern wie dem PIC oder 8051 (REFs) konzipierten Systemen auch Plattformen an, die explizit für den Anwendungszweck der Ansteuerung von Sensoren oder Aktuatoren und der Kommunikation mit einem Basissystem gefertigt werden. Exemplarisch kann hier die Smart-ITs-Plattform (REF) angeführt werden, die neben der flexiblen Ansteuerbarkeiten von unterschiedlichen Sensoren bereits Module zur Vernetzung

### 3 Input & Interpretation

untereinander und mit zentralen Diensten in der Infrastruktur anbietet.

Aus den eben angeführten Gründen erscheint zur Datenübertragung eine drahtlos arbeitende Technologie am geeignetsten. Aufgrund der geringen benötigten Reichweite und der Anforderung, möglichst energieeffizient zu arbeiten, bieten sich die Technologien "Bluetooth" und "ZigBee" an. Bluetooth erreicht höhere Übertragungsraten, ist aber in der Anzahl der gleichzeitig verwendbaren Geräte (max. 7) für den hier vorgestellten Anwendungsfall zu beschränkt. Ein ZigBee-Netz kann mit bis zu 255 Geräten gleichzeitig arbeiten und ist außerdem im Einsatz energiesparender. Für den gegebenen Anwendungsfall erschien also ZigBee als geeignete Technologie (und wurde auch bereits in (AON Cube, Simon Vogl REF) in einem ähnlichen Anwendungsfall erfolgreich eingesetzt).

Zur Feststellung des Öffnungsstatus eines Container-Tokens bieten sich bei aktiven Sensortechnologien mehrere Möglichkeiten an. Der Einsatz eines Schaltelements, das beim Öffnen den Kontakt herstellt oder unterbricht, erscheint als eine nahe liegende Lösung. Auch der Einsatz eines Drehelements am Angelpunkt des Öffnungsschaniers, dessen elektrische Eigenschaften (z.B. Widerstand oder Kapazität) mit dem Öffnungswinkel ändern, kann angedacht werden. Damit ist nicht nur eine Unterscheidung zwischen "offen" oder "geschlossen" sondern auch die Identifikation von Zwischenzuständen möglich.

Der Inhalt eines Container-Tokens kann ebenfalls mit unterschiedlichen Technologien erfasst werden. Die Zielsetzung ist hier nicht der Positionsbestimmung der eingebetteten Tokens sondern lediglich die Feststellung derer Identität. Naheliegend ist hierzu der Einsatz von elektromagnetischen Ansätzen wie oben beschrieben. Durch das Anbringen von z.B. RFID-Chips an den einzubettenden Tokens sowie eines Lesegeräts im Container-Token kann die Identifikation robust durchgeführt werden. Alternativ bieten sich Systeme an, die auf Gewichtsmessung basieren. Über einen in das Containertoken eingebauten Sensor wird dabei das Gesamtgewicht der eingebetteten Tokens bestimmt. Bei entsprechender Konzeption der einzubettenden Tokens (unterschiedliche Gewichte) kann aus dem Gesamtgewicht auf die tatsächlich enthaltenen Tokens geschlossen werden. Ein Nachteil dieses Ansatzes ist die beschränkte Anzahl von Tokens und die notwendige exakte Fertigung jedes einzelnen Tokens, da es bei Gewichtsabweichungen zu Fehlerkennungen kommt.

**Bewertung** Hinsichtlich der erreichbaren Flexibilität und zu erwartenden Servicequalität wäre in diesem Abschnitt eine Entscheidung zugunsten aktiver Tokens zu treffen. Im Gesamtkontext betrachtet und unter Berücksichtigung der Entscheidung für optische Systeme zur Bestimmung der Positionsparameter ist diese Wahl jedoch zu relativieren. Wie oben beschrieben, erlaubt eine auf optischen Systemen

beruhende Infrastruktur grundlegend die Umsetzung der geforderten Funktionalität. Gleichzeitig wird die Komplexität des Systems massiv reduziert und die Erstellung zusätzlicher Tokens vereinfacht (da keine zusätzliche Elektronik notwendig ist). Der Wegfall von Energieversorgung und Sensorlogik in den Token reduziert deren Gewicht und ermöglicht gleichzeitig mehr Platz für einzubettende Tokens.

Für den hier beschriebenen Anwendungsfalls bzw. die prototypische Umsetzung des Werkzeugs wird deshalb auf aktive Tokens verzichtet und der Einsatz von passiven Tokens bevorzugt. Der Mehraufwand in Erstellung und Wartung des Systems beim Einsatz aktiver Tokens wiegt in der Gesamtheit betrachtet die zu erwartende höhere Erkennungsqualität nicht auf.

#### **3.1.2 In Frage kommende Frameworks**

Unter Anbetracht der im vorherigen Abschnitt getroffenen grundlegender Technologieentscheidung zugunsten optischer Erkennungstechnologie mit passiven Tokens werden nun unterschiedliche Frameworks betrachtet, die die Umsetzung dieses Ansatzes erlauben. Es sind dabei zwei Klassen von Frameworks zu unterscheiden. *Generische Frameworks für Tangible Interfaces* beschäftigen sich generell mit dem zur Verfügung stellen von Services, die Kopplung von Sensoren und Aktuatoren mit Interpretations-Logik und letztendlich konkreten Applikationen erlauben. Sie gehen dabei nicht auf konkrete Sensortechnologie (wie die hier verwendeten optischen Ansätze) ein sondern versuchen eine Abstraktionsebene einzuführen, die die Applikationen von der konkreten Technologie entkoppelt und damit flexibler macht. *Frameworks für video-basierten Input für Tangible Interfaces* sind hochspezialisierte Produkte, die konkret für die Umsetzung von optischen Ansätzen zur Eingabe von Daten bei Tangible Interfaces entwickelt werden. Ihr Vorteil liegt im durch die Spezialisierung im Allgemeinen geringeren Aufwand zur Einrichtung und auch während des Betriebs. Echtzeit-Anforderungen sind oft nur mit spezialisierten Frameworks zu erreichen. Eine Kombinationsmöglichkeit zwischen Produkten der beiden Kategorien ergibt sich beim Einsatz eines spezialisierten Frameworks als Eingabe-Modul für eine generisches Framework. Durch diesen Ansatz kann die einfache Inbetriebnahme spezialisierter Frameworks mit der Flexibilität generischer Frameworks zusammengeführt werden.

##### **Generische Frameworks**

Generische Frameworks zur Behandlung von Input und Output bei Tangible Interfaces sind historisch nicht exakt von anderen Frameworks abzugrenzen, die im Umfeld des Ubiquitous bzw. Pervasive Computing entwickelt wurden. Derartige Ansätze wurden erstmals im Zusammenhang mit "Context Computing" erwähnt, um Ap-

### 3 Input & Interpretation

plikationen eine generische Möglichkeit zu bieten, Information aus der Umgebung über beliebige Sensoren zu erfassen und diese zu aggregieren und zu interpretieren. Aufbauend auf dieser Interpretation sollen Aussagen über den aktuellen Zustand der Umgebung (den "Kontext") getroffen werden können, die diese Applikationen zur Adaption benutzen können. Der Rückkanal, also die Ansteuerung von Aktuatoren, wurde erst in späteren Entwicklungen berücksichtigt. Die meisten Systeme dienen explizit nicht der Erstellung von marktreifen Applikationen sondern widmen sich eher der Umsetzung von "Rapid Prototyping"-Ansätzen im Bereich der Tangible Interfaces. Begründet wird dies mit der oft suboptimalen Ressourcen-Ausnutzung, die mit der Generalisierung und Flexibilisierung des Frameworks einhergeht.

Die Aufzählung der hier beschriebenen Frameworks erhebt keinen Anspruch auf Vollständigkeit. Es wurde eher darauf geachtet, historische bzw. für den konkreten Anwendungsfall geeignete Ansätze aufzunehmen und in ihren wesentlichen Eigenschaften zu beschreiben.

**Context Toolkit** Das Context Toolkit (Dey et al., 2001) ist das historisch erste Framework, das versucht, die starre Verbindung zwischen Sensoren und Applikationslogik aufzubrechen und eine konfigurierbare Schicht einzuziehen, die eine schnellere, generischere Applikationsentwicklung ermöglicht und die Wiederverwendbarkeit einmal entwickelter Komponenten erhöht.

Konzeptuell existieren im Framework drei Arten von Komponenten: Context Widgets, Context Interpreters und Context Aggregators. Context Widgets implementieren die Ansteuerung beliebiger Software- und Hardwaresensoren und sind für das Sammeln von Information über die Umgebung zuständig. Sie vermitteln zwischen der physischen Umgebung und den konzeptuell höher liegenden Komponenten indem sie die unverarbeiteten Kontextdaten mittels einer geeigneten Schnittstelle kapseln und bestimmte Funktionen zur ersten Auswertung der Rohdaten ausführen. Eine Anwendung kann diese Daten verwenden, ohne dass sie Detailkenntnisse über die zugrunde liegenden Sensortechnologien haben muss. Context Interpreter aggregieren die Sensordaten zu komplexeren Kontextinformationen d.h. sie konvertieren und interpretieren die Daten mehrerer Context Widgets und versuchen diese zu einheitlichen Clustern zusammenzufassen. Context Aggregators dient der Zusammenführung verschiedener Kontextinformationen die für bestimmte Anwendungen relevant sind. Context Aggregators bilden damit die Schnittstelle zu den eigentlichen Applikationen.

Das Context Toolkit bietet nicht nur eine Softwareschnittstelle zu physischen Sensoren, es trennt auch die Akquisition und Repräsentation von der Auslieferung der Daten an kontextsensitive Applikationen.

**SiLiCon Context Framework** Das SiLiCon Context Framework (Beer et al., 2003) ist ein Vertreter jener Klasse von Frameworks, deren Verhalten zur Laufzeit dynamisch konfigurierbar sind. Dies bedeutet im konkreten Fall, dass Applikationen die auf Basis des SiLiCon Context Framework erstellt wurden ihr Verhalten und ihren Aufbau aufgrund eintretender Ereignisse verändern können. Dies betrifft sowohl das Aktivieren und Deaktivieren von Input- und Output-Kanälen also auch die Interpretation der eingehenden Information und die Reaktion darauf. Das Framework wurde entworfen, um Szenarien zu beschreiben, in denen interaktive Systeme kontextsensitiv - d.h. abhängig vom aktuellen Zustand ihrer Umwelt - reagieren müssen.

Die grundlegenden Bausteine des SiLiCon Context Framework sind "Entitäten", die Objekte der realen Welt konzeptuell abbilden. Diese "Entitäten" besitzen "Attribute", also Eigenschaften, mit Hilfe derer die Entität näher beschrieben wird. Über "Attribute" kann die Wahrnehmung einer Entität von deren Umwelt sowie deren Interaktionsmöglichkeiten mit derselben beschrieben werden. Mit Hilfe von "ECA"-Regeln (*Event-Condition-Action*) wird beschrieben, auf welche Wahrnehmung der Umwelt (Event) eine Entität unter welchen Bedingungen (Condition, formuliert auf Basis des internen Zustands der Entität) mit welchen Aktivitäten (Action) reagiert. Diese Regeln können zur Laufzeit dynamisch verändert und nachgeladen werden. Außerdem ist es möglich, in "Actions" das Nachladen von Entitäten oder das Hinzufügen oder Entfernen einzelner Attribute durchzuführen (Oppl, 2004, S. 90).

Das SiLiCon Context Framework abstrahiert durch seine konzeptionelle Struktur mit dem Einsatz von "Entitäten" und "Attributen" nicht so stark von der realen Welt wie der Context Toolkit Ansatz - die Abbildung ist im ersten Schritt "direkter" und muss erst im zweiten Schritt technisch konkretisiert werden, ein klassischer Softwareengineeringprozess (im Sinne von "Analyse - Design - Implementierung") wird damit vollständiger (auch in den ersten Phasen) durch das Framework abgebildet und unterstützt.

**Papiermaché** Eines der ersten Rapid-Prototyping Frameworks

**TUIpist** Das TUIpist-Framework (Furtmüller, 2007) wurde im Zusammenhang mit der hier vorgestellten Arbeit entwickelt (Furtmüller and Oppl, 2007). TUIpist verfolgt einen ähnlich modularen Ansatz wie die anderen hier vorgestellten Frameworks, setzt jedoch zur Koordination der Module untereinander einen datenzentrierten Ansatz - auf dem LINDA-Konzept (REF) beruhende TupleSpaces - ein. Die konzeptionelle Modulstruktur ist ähnlich der Aufteilung, die bereits von Dey et al. (2001) im Context Toolkit vorgeschlagen wurde.

Die grundlegenden Module, die im Framework verwendet werden, sind "Senso-

### 3 Input & Interpretation

ren", "Aggregatoren" und "Anwendungen / Aktuatoren" (siehe Grafik XY 1). "Sensor"-Module binden externe Datenquellen an das Framework an. Sie enthalten dazu eine sensor-spezifische Komponente, die die Schnittstelle zur jeweiligen Hard- bzw. Software bildet. Über diese Schnittstelle gelieferte Daten werden in einer zweiten Komponente vorverarbeitet und soweit abstrahiert, dass die Datenrepräsentation unabhängig von der die Daten liefernden Sensortechnologie ist (z.B. Abbildung von GPS-Koordinaten auf logische Positionsinformation, die auch aus anderen Quellen stammen könnte). "Aggregatoren" fassen die Information mehrerer "Sensor"-Module zusammen und interpretieren ggf. einander ergänzende oder auch widersprechende Information. Sie werden immer aktiv, wenn neue Sensordaten zur Verfügung stehen und aktualisieren dabei die Information über den Gesamtzustand der den Framework bekannten Umwelt. "Anwendungen / Aktuatoren" bilden letztendlich die Schnittstelle zu konkreten Applikationen, die ihr Verhalten an den aktuellen Umweltzustand anpassen bzw. diesen darstellen oder Aktuatoren, die basierend auf dem aktuellen Umweltzustand Aktionen in dieser setzen. "Anwendungen / Aktuatoren" filtern dabei wieder den von "Aggregatoren" gelieferten Gesamtzustand der bekannten Umwelt und liefern nur die für die Applikation relevanten Daten aus. Dabei kann erneut eine Nachverarbeitung der Daten, z.B. im Sinne einer Anpassung an eine externe Schnittstelle, erfolgen.

Die Verbindung der Komponenten erfolgt wie erwähnt mittels einem daten-zentrierten Ansatz. Eine wesentliche Eigenschaft dieses Ansatzes ist, dass keine explizite Verknüpfungen einzelner Module definiert werden. Die Zuordnung erfolgt vielmehr indirekt durch die Daten selbst. Jedes Modul kann Datensätze (Tupel) in einem definierten Format generieren und in einen gemeinsam genutzten Datenraum - den Tuplespace - stellen. Andere Module können nun Anfragen an den Tuplespace stellen, ob Daten, deren Struktur oder Inhalt gewissen Kriterien entspricht, vorhanden sind. Ist dies der Fall, können diese Daten aus dem Tuplespace entnommen werden und nach erfolgter Verarbeitung in modifizierter Form wieder eingestellt werden (siehe Abbildung XY). Das Tuplespace-Konzept erlaubt auch eine dynamische Erweiterung bzw. Veränderung sowohl der Ein- und Ausgabekanäle als auch der internen Datenverarbeitung zur Laufzeit, indem zusätzlich Module am Tuplespace registriert werden. Durch die lose Koppelung der Module muss keine zusätzliche Konfiguration an anderen Modulen oder am Tuplespace selbst vorgenommen werden.

Die Implementierung von TUIpist auf Basis des Java Jini-Frameworks (REF) ermöglicht eine Verteilung der einzelnen Module einer Applikation auf unterschiedliche Rechner (im Sinne eines "verteilten Systems" (REF)) ohne zusätzlich vom Implementierer zu leistenden Koordinationsaufwand. Es können damit auch (räumlich) entfernte Sensoren oder Webapplikationen angebunden werden und der ggf. auftretende Rechenaufwand zur Aggregation oder Interpretation von Daten auf mehrere Rechner verteilt werden.

- Grafiken aus TICE Paper? -

#### **Frameworks für video-basierten Input**

Im Gegensatz zu den eben beschriebenen generischen Frameworks wurden die hier vorgestellten Frameworks explizit für die Behandlung von video-basiertem Input entwickelt. Wie oben bereits erwähnt stehen die beschriebenen Frameworks mit obigen insofern in Zusammenhang, also dass sie zumeist als Sensor-Komponente in generischen Ansätzen eingesetzt werden können. In ihrer grundlegenden Ausrichtung sind Sie jedoch für den unmittelbaren Einsatz in einer Endanwendung konzipiert.

Die hier vorgestellten Systeme implementieren den Ansatz des code-basierten optischen Trackings. Feature-basierte Ansätze kommen im konkreten Anwendungsfall nicht in Frage, da zur Modellierung eine Vielzahl von gleichartigen Objekten eingesetzt wird, die sich in ihrem äußereren Erscheinungsbild nicht unterscheiden und damit in feature-basierten Ansätzen nicht eindeutig identifiziert werden können.

Wie bereits im letzten Abschnitt erhebt auch die hier angeführte Aufzählung keinen Anspruch auf Vollständigkeit. Neben der Darstellung der historischen Entwicklung des Feldes und der Beschreibung von in Forschung und Praxis relevanten Ansätzen wurde bei der Auswahl vor allem auf freie Verwendbarkeit und Zugriff auf den Source-Code der Erkennungsroutinen geachtet, da die Notwendigkeit applikationsspezifischer Anpassungen nicht auszuschließen war (etwa um benötigte aber nicht direkt unterstützte Parameter zu extrahieren).

#### **ARToolkit** Historisch erstes Framework, Mapping, beschränkter Code Raum

**Visual Codes** Das Visual Codes System ist ein Vertreter der direkt codierenden Ansätze, d.h. dass die Nutzinformation ohne Zwischenschritt direkt aus dem Code extrahiert werden kann. Im Gegensatz zu den standardisierten und kommerziell genutzten Code-Formate QR-Code (REF) oder Datamatrix (REF), die eine Kapazität von bis zu einigen hundert Byte haben, bietet das Visual Code System lediglich 80 Bits an Nutzinformation an, was dazu führt, dass in vielen Anwendungsfällen ein Mapping durch die Anwendung durchgeführt wird, um die zu verwaltende Information abbilden zu können.

– Abb. Visual Codes –

Der Vorteil des Visual Code Systems liegt in seiner mächtigen Auswertbarkeit bei gleichzeitig geringem Bedarf an Rechenkapazität. In der Standardimplementierung werden neben der Position und der Rotation in der Ebene auch die Neigungspa-

rameter im Raum extrahiert. Der Erkennungsalgorithmus läuft dabei in Echtzeit und kann unverändert sogar auf Java-fähigen Mobiltelefonen ausgeführt werden. Durch die relativ geringe Datendichte der Codes reichen dementsprechend auch verhältnismäßig niedrig auflösende Bilder (z.B. 320 x 240 Bildpunkte) für eine Erkennung aus. Wie alle anderen optischen Ansätzen leidet das Visual Code System unter Erkennungsproblemen bei wechselnden Lichtverhältnissen und insbesondere bei schlechtem Kontrast. Diese Verhalten kann in der vorliegenden Implementierung auch nicht durch Rekonfiguration korrigiert werden.

Das Visual Code System muss von auf ihm aufbauenden Applikationen direkt auf Source-Code-Ebene eingebunden werden, eine vorgegebene externe Schnittstelle existiert nicht. Anbindungs Routinen für Kameras existieren nur für Mobiltelefone und müssen für andere Plattformen ggf. neu erstellt werden.

**ReacTIVision** ReacTIVision (REF) ist ein frei verfügbares Framework zu optischen Erkennung von Codes in Echtzeit. ReacTIVision arbeitet mit proprietären Codes (siehe Abbildung 3.1), in denen die Information nicht an bestimmte Positionen sondern im Wesentlichen in die Anzahl und Schachtelung der Schwarz-Weiß-Übergänge codiert ist. Die Mächtigkeit der Informationscodierung direkt in den Code ist damit beschränkt, es wird deswegen ein Mapping-Ansatz verwendet, um die eigentliche Nutzinformation auf Codes abzubilden. Grundsätzlich wäre jedoch die direkte Codierung einer beschränkten Anzahl von Bits möglich, ist aber nicht unmittelbar vorgesehen.

Durch die Art der Informationscodierung ist die Erkennungsleistung von ReacTIVision auch unter schlechten Lichtbedingungen oder bei verzerrtem Eingangsbildern akzeptabel bis sehr gut. Die Form der Codes ist außerdem nicht vorgegeben, sie kann frei gewählt werden. Sogar händisch gezeichnete Codes können erkannt werden, da ausschließlich eine geschlossene Außenlinie und entsprechende Schwarz-Weiß-Übergänge innerhalb dieser Linie erfassbar sein müssen.



Abbildung 3.1: ReacTIVision Code

Die ReacTIVision-Software ist plattformübergreifend für Windows, Linux und Mac OS X verfügbar. Sie greift über plattformspezifische Schnittstellen auf angeschlossene Kamera zu und wertet das empfangene Bild in Echtzeit aus. Dabei sind

bei einer Kameraauflösung von 1024 x 68 Bildpunkten Bildraten von 15-20 Bildern pro Sekunde erreichbar. Diese Leistung wird auch durch eine höhere Anzahl von gleichzeitig im Bild vorhandenen Codes nicht merklich geringer.

Neben der Position der Codes wird auch deren aktuelle Rotation sowie die erste Ableitung dieser drei Parameter (also ein Maß für die Bewegung) extrahiert. Zudem können Finger, die die Oberfläche berühren erkannt und deren Position extrahiert werden. Dies ist auch für mehrere Finger möglich, wobei eine eindeutige Zuordnung über die Zeit erhalten bleibt und so rudimentäre Multitouch-Funktionen umgesetzt werden können.

Als problematisch stellt sich wie auch bei allen anderen betrachteten optischen Ansätzen die Abhängigkeit der Erkennungsqualität von der Umgebungsbeleuchtung bzw. deren Änderung dar. ReacTIVision arbeitet mit adaptiven Filteralgorithmen zur Aufbereitung des Bildes, was jedoch nur leichte Beleuchtungsschwankungen ausgleichen kann. Die Software kann händisch an die Beleuchtungsverhältnisse angepasst werden (Einstellung der Blendenöffnung und der Bildverstärkung), bei sich ändernden Lichtverhältnissen muss jedoch regelmäßig eine manuelle Nachführung der Parameter vorgenommen werden.

Die aus dem Bilderstrom gewonnenen Daten werden im Falle einer Änderung zu mindest eines Wertes über ein Netzwerkschnittstelle (UDP-basiert) in einem proprietären Protokoll zur Verfügung gestellt. Zu diesem Protokoll werden Schnittstellen und Referenzimplementierungen in unterschiedlichen Programmiersprachen, unter anderem C(++) und Java, zur Verfügung gestellt. Applikationen können diese Schnittstelle implementieren und werden sie mittels insgesamt sechs zu implementierenden Methoden an die Erkennungsroutinen angebunden (3 für Code- und 3 für Fingertracking).

#### **3.1.3 Technologieentscheidung**

Auf Basis der Anforderungen, die im Anwendungsfall an das Werkzeug gestellt werden, ist nun nach der grundsätzlichen Entscheidung für ein auf optischen Ansätzen basierenden System die Entscheidung für ein konkretes Framework zu treffen.

#### **Vergleich der Frameworks für videobasierter Input**

Für die Anbindung von videobasiertem Input wurde drei Frameworks vorgestellt. Diese Frameworks sind nun hinsichtlich mehrere Aspekte zu vergleichen, die sich aus den Anforderungen der zu erstellenden Applikation sowie der Forderung nach möglichst einfacher (im Sinne von unaufwändiger) Einbindung des Frameworks ergeben. Im Einzelnen sind dies

### 3 Input & Interpretation

- die Unterstützung bei der Einbindung von Kameras, eine Schnittstelle zur Programmiersprache Java,
- eine stabile und in Echtzeit ablaufende Bilderkennung,
- eine ausreichende Anzahl von Codes (Größenordnung 100),
- die Extraktion von Position und Rotationsinformation für Codes im Erfassungsbereich der Kamera sowie
- die technische und lizenzerrechtliche Möglichkeit, die Frameworks an die eigenen Anforderungen anzupassen.

ReacTIVision bietet die umfassendste Unterstützung zur Einbindung unterschiedlicher Videoquellen und ist zur Anwendungsseite hin durch die Netzwerkschnittstelle am flexibelsten einsetzbar. Alle drei Ansätze bieten Schnittstellen bzw. Konnektoren für die Programmiersprache Java an. Die Frameworks selbst sind in C(++) oder Java erstellt und können auf den gängigen Plattformen (Windows, Linux, Mac OS x) ausgeführt werden. Eine Verteilung der Applikation auf mehrere Rechner wird nur von ReacTIVision explizit unterstützt.

Hinsichtlich der eigentlichen Bilderkennungs-Routinen sind die Ansätze in ihrer Leistung und Geschwindigkeit vergleichbar, leiden aber alle unter der Abhängigkeit von der Umgebungsbeleuchtung. ReacTIVision bietet hier als einziger Ansatz die Möglichkeit, Kameraparameter zur Laufzeit nachzuführen und so Schwankungen der Umgebungshelligkeit auszugleichen.

Durch den eingesetzten Mapping-Ansatz sind ARToolkit und ReacTIVision hinsichtlich Form und Inhalt der Codes flexibler als direkt codierende Ansätze wie Visual Codes. Dies ist im konkreten Anwendungsfall relevant, da aufgrund der Token-Form und deren beschränkter Größe eine ideale Platzausnutzung erfolgen muss, um ausreichende Erkennungsleistung zu gewährleisten.

Die Extraktion der Raumparameter (Neigungsinformation, ...), die von ARToolkit und Visual Codes ermöglicht wird, ist in bei tisch-basierten Werkzeugen wie dem hier vorgestellten nicht notwendig - die Erhebung planarer Parameter (Position und Rotation) ist ausreichend.

Die Anzahl der verfügbaren und zuverlässig unterscheidbaren Codes muss für die hier vorgeschlagene Anwendung in der Größenordnung 100 liegen. Visual Codes und ReacTIVision erfüllen diese Anforderung, ARToolkit bietet im Lieferumfang weniger Codes an, diese können jedoch erweitert werden und bieten auch in der geforderten Anzahl noch ausreichend Unterscheidungsmerkmale für eine zuverlässige Unterscheidung (REF Schmalstieg TU System).

Von allen drei Systemen steht der Source-Code zur Verfügung. Lizenzrechtlich erlauben ebenfalls alle Systeme eine Veränderung des Sourcecode (LIZENZEN AUF-ZÄHLEN)

### 3.1 Möglichkeiten zur Erfassung von Benutzerinteraktion

---

	ARToolkit	Visual Codes	ReacTIVision
Kamera-unterstützung	?	nativ nur für Mobiltelefone	nativ auf allen unterstützten Plattformen
Plattformen	x	Java-basiert auf allen Plattformen	Windows, Linux, Mac OS X
Schnittstelle zu Java	ja	ja	ja
Stabilität der Bilderkennung	ehrer hoch	niedrig	hoch
Geschwindigkeit der Bilderkennung	> 10 fps	> 10 fps	> 10 fps
Anzahl von Codes	niedrig	hoch	hoch
Erkennung von Position	ja	ja	ja
Erkennung von Rotation	ja	ja	ja
Source-Code verfügbar	ja	ja	ja
Lizenz	?	?	?

Auf Basis dieses Vergleichs ist erkennbar, dass das ReacTIVision-Framework für den hier verfolgten Ansatz am besten geeignet erscheint. Es wird deshalb zur Umsetzung des Werkzeugs verwendet. Der zusätzliche Einsatz eines generischen Frameworks zur Flexibilisierung der Applikationsstruktur ist damit nach wie vor möglich und wird im nächsten Abschnitt diskutiert.

### Vergleich der generischen Frameworks

Der Einsatz eines generischen Frameworks ist im konkreten Anwendungsfall für die Modularisierung der Interpretations- und Output-Module denkbar. Eine weitere Anwendung von generischen Frameworks ist die Anforderung nach der Skalierbarkeit der Anzahl der Ausgabemodule ggf. auch während des Betriebs der Applikation (um z.B. weitere Viewer-Module zur Beobachtung des Modellierungsvorganges während der Laufzeit zuschalten zu können). Die Aspekte die beim Vergleich der vorgestellten Ansätze berücksichtigt werden müssen, sind deshalb

- die Unterstützung von modularen funktionalen Einheiten sowie

### 3 Input & Interpretation

- die Möglichkeit diese zur Laufzeit zu laden und entfernen und
  - die Konfigurierbarkeit der Verknüpfungen dieser Module.
  - Daneben sind nicht-funktionale Anforderungen wie Skalierbarkeit und
  - Effizienz bei der Weiterleitung und Verteilung der Anwendungsdaten
- zu berücksichtigen.

Die Modularisierung der Applikation wird von allen vier vorgestellten Frameworks unterstützt. Das Context Toolkit, Papiermaché und TUIpist unterscheiden dabei konzeptuell zwischen unterschiedlichen Arten von Modulen (im Wesentlichen "Input", "Verarbeitung" und "Output"), das SiLiCon Context Framework unterscheidet nicht zwischen unterschiedlichen Modultypen, dort wird die Rolle eines Moduls ausschließlich über seine Attribute (also die nach außen sichtbaren funktionalen Einheiten) definiert.

Eine Erweiterbarkeit der Applikation zur Laufzeit ist nur beim SiLiCon Context Framework und bei TUIpist möglich. DAs SiLiCon Context Framework arbeitet hier mit eigens erstellten Java-Classloadern, die das Nachladen von Klassen (Modulen) und deren Integration in die Infrastruktur erlauben. TUIpist verwendet die inhärent dynamische Erweiterbarkeit des Java Jini (REF) Frameworks, auf Basis dessen es implementiert wurde. Das Context Toolkit und Papiermaché erlauben kein Nachladen oder Entfernen funktionaler Einheiten während der Laufzeit.

Die Konfiguration der Verbindungen zwischen den Modulen ist bei allen Frameworks möglich, konzeptuell jedoch unterschiedlich ausgeführt. Im Context Toolkit und in Papiermaché muss die Verbindung direkt im Programmcode codiert werden und wird im wesentlichen auf Methodenaufrufe abgebildet. Das SiLiCon Context Framework verwendet ereignisbasierte Regeln, um Module zu verknüpfen. Ein von einem Modul ausgelöstes Ereignis kann hier (ggf. durch Bedingungen eingeschränkt) Aktionen in anderen Modulen auslösen. Eine Besonderheit dieses Ansatzes ist, dass Teile der Interpretationslogik (also der Erkennung von Benutzerinteraktion aus den Rohdaten) in die Regeln ausgelagert werden und damit jederzeit dynamisch verändert werden können. So kann zum Beispiel die Interpretation der Nähe zwischen zwei Token in einer Regel codiert werden und so in die Reihe der zulässigen (bzw. erkennbaren) Interaktionen aufgenommen werden. TUIpist benötigt hingegen keinerlei explizite Verbindung der Module, da sämtliche Koordination über den geteilten Datenraum ausgeführt wird. Die Verknüpfungslogik wird hier in die Module verschoben, die selbst wissen müssen, für welche Daten für sie ggf. relevant sind und welche sie deshalb dem Tuplespace entnehmen.

Hinsichtlich der Skalierbarkeit der Frameworks können aufgrund mangelnder Vergleichsdaten keine fundierten Aussagen getroffen werden. Hinzuweisen ist jedoch auf die Unterstützung von verteiltem Betrieb einer Applikation durch das SiLiCon

Context Framework (via SOAP-Aufrufe (REF) und das TUIpist Framework (via Jini und Java RMI (REF)). Durch diese Verteilbarkeit ist kann die Problematik mangelnder Rechenressource umgangen werden, die vor allem bei rechenintensiven Anwendungen wie der eingesetzten Bildanalyse im optischen Tracking auftritt.

Die Beurteilung der Effizienz der Frameworks ist hier in Hinblick auf die geforderte Echtzeit-Interaktion mit dem System als relevant zu erachten. Das Haupt-Kriterium für Effizienz ist dementsprechend die Verzögerung zwischen Eingabe-Ereignissen und dem Feedback auf Applikationsebene, die beim Einsatz der Frameworks auftritt. Eine Beurteilung in absoluten Zahlen kann hier mangels entsprechenden Datenquellen ebenfalls nicht erfolgen, konzeptuell sind aber bei den Frameworks, in denen Module direkt durch Programm-Code verknüpft werden (Context Toolkit und Papiermaché), eher geringe Verzögerungen zu erwarten. Die beiden Frameworks, die mit expliziter und konfigurierbarer Middleware zur Datenvermittlung arbeiten (SiLiCon Context Framework und TUIpist) lassen eher höhere Verzögerungen erwarten, wobei der ereignisbasierte Ansatz des SiLiCon Context Framework dem daten-basierten Ansatz von TUIpist insofern überlegen ist, als das bei der datenbasierten Interaktion die Module in regelmäßigen Intervallen nach neuen Daten suchen müssen (Polling, Information Pull-Ansatz), während bei ereignisbasierten Ansätzen die Benachrichtigung der betroffenen Module automatisch und zum Zeitpunkt des Auftretens des Ereignisses erfolgt (Information Push-Ansatz). Der Information Pull-Ansatz sorgt dabei einerseits für erhöhten Kommunikationsaufwand und potentiell für Verzögerungen bei Ereignissen, die innerhalb eines Polling-Intervalls auftreten.

#### **- VERGLEICHSTABELLE -**

Auf Basis dieser Gegenüberstellung erscheinen das SiLiCon Context Framework und TUIpist als die beiden geeignetsten Kandidaten für den Einsatz als generisches Framework im hier vorgestellten Anwendungsfall. Das Context Toolkit und Papiermaché scheiden aus verschiedenen Gründen, vor allem aber der mangelnden Flexibilität aus.

Stellt man die beiden verbleibenden Frameworks gegenüber, so sind hinsichtlich der funktionalen Anforderungen keine Vorteile für einen der beiden Kandidaten zu identifizieren. Hinsichtlich der nichtfunktionalen Anforderungen scheint TUIpist hinsichtlich der Skalierbarkeit leichte Vorteile zu haben, da das Nachladen von neuen Instanzen weniger Aufwand versacht als im Fall des SiLiCon Context Framework (lediglich Laden eines Moduls im ersten Fall gegenüber Laden und Einbinden über eine Änderung der Regelbasis im zweiten Fall). Außerdem ist die technologische Basis der Verteilungsarchitektur in TUIpist konzeptionell auf höherer Ebene angesiedelt und mächtiger in der Verwaltung der Applikationsstruktur (unter anderem werden neue Module bei TUIpist automatisch in die Infrastruktur eingebun-

den, sobald sie angemeldet werden, im SiLiCon Context Framework muss für jeden Kommunikationskanal die IP-Adresse des Empfängers bekannt sein und in die Regelbasis aufgenommen werden).

Hinsichtlich der Effizienz der Kommunikation bietet das SiLiCon Context Framework gegenüber TUIpist aus den oben beschriebenen Gründen (Benachrichtigung über Änderungen gegenüber Polling) Vorteile. Für die hier beschriebene Applikation fällt die Entscheidung trotzdem zugunsten TUIpist. Neben der generell unaufwändigeren Konfigurierbarkeit kommt die Struktur von TUIpist einem iterativen Software-Entwicklungsprozess insofern eher entgegen, als dass die Modularisierung von initialen, monolithischen Prototypen (z.B. basierend auf der Struktur des zuvor ausgewählten ReacTIVision-Frameworks) einfacher möglich ist. Dies liegt darin begründet, dass in der modularen und dynamischen Struktur von TUIpist die gesamte Applikationslogik in den Modulen enthalten ist und so Teile aus einer monolithischen Applikation herausgelöst und direkt übernommen werden können, wobei lediglich die Logik der Datenübergabe von direkten Methoden-Aufrufen auf die TUIpist-Routinen zum Zugriff auf den Tuplespace umgearbeitet bzw. erweitert werden muss. Beim korrekten Einsatz des SiLiCon Context Framework wandert wie oben beschrieben ein Teil der Applikationslogik in die Regelbasis, was erhöhten Aufwand bei der iterativen Entwicklung verursacht und außerdem das Zusammenspiel der Applikations-Module unübersichtlicher und schwerer fassbar macht.

## **3.2 Konzeption und Umsetzung der Hardwarekomponenten**

Basierend auf der oben getroffenen Techhnologieentscheidung zugunsten eines optischen, video-basierten Input-Systems für das hier zu erstellende Tabletop Interface wird in diesem Abschnitt das konkrete Hardware-Design beschrieben. Dies umfasst die Beschreibung des Tabletop Interfaces im Überblick und detaillierte Betrachtungen des Token-Designs sowie der Tisch-Oberfläche, die als Input-Kanal dient. Im weiteren werden spezifische Herausforderungen und der jeweilige hier verfolgte Lösungsansatz beschrieben. Nicht Gegenstand dieses Abschnitts sind jene Hardware-Komponenten, die für die Ausgabe von Information eingesetzt werden. Obwohl hier im Überblick angeführt, werden sie detailliert erst in Kapitel XY über die Visualisierung der Modellierungsinformation beschrieben.

### **3.2.1 Überblick**

### 3.2 Konzeption und Umsetzung der Hardwarekomponenten

Das Tabletop Interface ist als Tisch mit einer Oberfläche von 100 cm x 80 cm ausgeführt. Die Höhe beträgt 110 cm. Die wesentlichen Hardwarekomponenten sind die Tischoberfläche, die in semi-transparenten Acrylglas ausgeführt ist und die Bodenplatte, in die die Kamera zum optischen Tracking der Tokens auf der Oberfläche sowie der Videoprojektor zur Ausgabe von Information auf der Oberfläche (Projektion von unten) eingebaut sind (siehe Abbildung 3.2). Der Tisch ist auf allen Seitenflächen mit Platten aus Styropor (MARKENNAME -> KUNSTSTOFF) verkleidet, um im Inneren kontrollierte Umgebungslichtbedingungen für die Bilderfassung mit der Kamera zu schaffen. Die kontrollierten Bedingungen werden durch den Einsatz von vier Beleuchtungsmodulen gewährleistet, die ebenfalls in der Bodenplatte integriert sind und über den Seitenflächen eingegebene Streuscheiben für eine einheitliche, diffuse Beleuchtung des Tischinnerenraums sowie der Oberfläche sorgen.

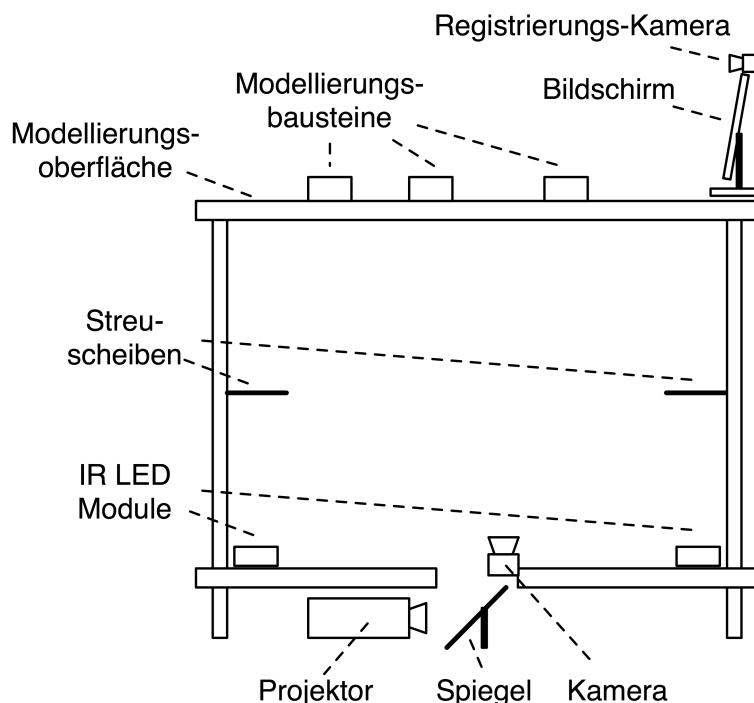


Abbildung 3.2: Überblick über den Aufbau des Tabletop Interfaces

Am Tisch befindet sich zusätzlich ein Bildschirm, auf dem entsprechend der Anforderungen zur Unterstützung der Modellierung Zusatzinformation ausgegeben wird. Eine zweite Kamera, die am Bildschirm sitzt und der unterstützenden Erfassung von Information dient (zum Beispiel der Registrierung von geschachtelten Token wie oben bereits beschrieben).

### 3 Input & Interpretation

Das gesamte System ist so zerlegbar, dass es im Kofferraum eines Mittelklassewagens transportiert werden kann. Es werden dazu die Verkleidungsplatten und Tischbeine entfernt, die Tischplatte kann dann direkt auf die Bodenplatte gesetzt und verbunden werden. Das Volumen reduziert sich dabei auf 100 cm x 80 cm x 20 cm, wobei die Tischbeine, die Verkleidungsplatten und der Videoprojektor separat transportiert werden müssen. Ein Zusammenbau des Systems inklusive Kalibrierung der Eingabe- und Ausgabe-Kanäle ist in etwa 30 Minuten möglich.

#### **3.2.2 Tokens & Input-Werkzeuge**

In diesem Abschnitt werden die einzelnen durch die Benutzer manipulierbaren Token-Ärten beschrieben. Neben den eigentlichen Modellierungstokens sind dies die Tokens zur Einbettung in Container sowie die Werkzeugtokens, von denen es Ausführungen zur generischen Interaktion mit dem System und Tokens zur Auslösung bzw. Kontrolle spezifischer Funktionalitäten gibt.

##### **Modellierungs-Tokens**

Die eingesetzten Modellierungs-Tokens sind aus nicht transparentem Acrylglas gefertigt. Ihre Außenmaße betragen in etwa (je nach Form) 10 cm x 6 cm in der Grundfläche und 4 cm in der Höhe. Damit ist einerseits eine ausreichende Größe zur Anbringung der ReacTIVision-Codes gewährleistet, andererseits sind noch klein genug, um in einer Hand gehalten und manipuliert werden zu können. Die Codes werden auf der Unterseite der Tokens angebracht und auch von unten erfasst (siehe nächster Abschnitt), um eine Verdeckung der Codes während der Manipulation zu verhindern (siehe Abbildung 3.3).

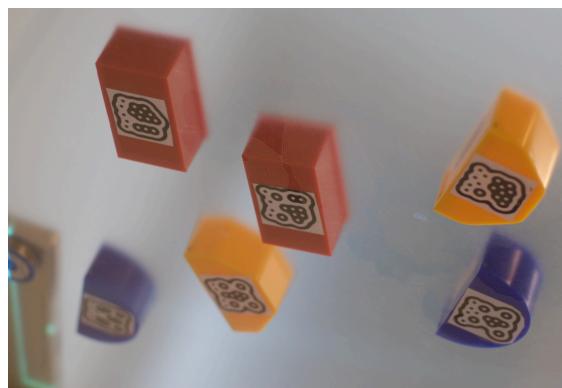


Abbildung 3.3: An Tokens angebrachte ReacTIVision-Codes zur Identifikation

## 3.2 Konzeption und Umsetzung der Hardwarekomponenten

---

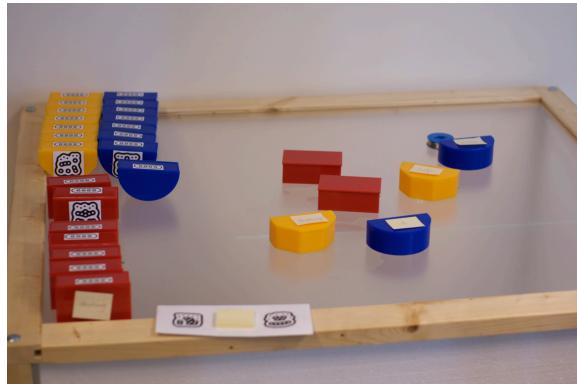


Abbildung 3.4: Arten von Modellierungstokens

Die Modellierungs-Tokens wurden in drei Ausführungen gefertigt (siehe Abbildung 3.4). Diese unterscheiden sich in Form und Farbe und können während der Modellierung von den Benutzern frei mit Bedeutung belegt werden. Die Auswahl der Formen und Farben erfolgte inspiriert von den Symbolen gängiger Modellierungsnotationen in Abstimmung mit fertigungstechnischen Einschränkungen. Eine wie in XY (REF von TEI) vorgesetzte Analyse geeigneter Token-Formen und eine auf diesen Ergebnissen basierende Umsetzung wurde nicht vorgenommen. Die liegt vor allem in der Tatsache begründet, dass sich der Fokus der hier vorgestellten Arbeit im Entwicklungsprozess von einem Werkzeug zur Geschäftsprozessmodellierung (mit vorgegebener Notation) hin zu einem allgemeiner einsetzbaren Werkzeug zur generischen Modellierung konzeptueller Modelle (ohne vorgegebene Notation) entwickelte. Die Auswahl der Token-Formen fiel in die erste Phase, wodurch die Tokens im äußeren Erscheinungsbild an gängige Notationen zur Ablauf-Modellierung angelehnt sind. Die Auswirkung der Token-Form auf die Modellierung scheint aber bei Benutzern ohne Modellierungs-Vorbildung geringen bis keinen Einfluss auf den Modellierungsprozess zu haben (siehe Kapitel XY - Evaluierung).

Wie in der Beschreibung der Anforderungen gefordert und oben bereits beschrieben sind die Modellierungs-Tokens als Container ausgeführt. Im Abschnitt über die Erkennung des Token-Zustands (REF) wurde beschrieben, dass beim Einsatz von optischem Tracking eine Möglichkeit geschaffen werden muss, im Kamerabild zu erkennen, ob ein Token geöffnet ist oder nicht. Um den konsequenten Einsatz eines Erkennungsframeworks (ReacTIVision) zu gewährleisten, wird zu diesem Zweck ein zweiter Code eingesetzt (siehe Abbildung 3.5), der nur dann für die Kamera sichtbar wird, wenn das Token geöffnet ist. Hardwareseitig ist dies so umgesetzt, dass die Modellierungstokens einen Öffnungsmechanismus besitzen, dessen Schanier nicht am Deckel sitzt (und damit nur diesen beweglich machen würde), sondern an der Bodenplatte, wodurch sich beim Öffnen eines Containers nicht nur der Deckel sondern

### 3 Input & Interpretation



Abbildung 3.5: Rückwand von Container Tokens

auch die Hinterwand des Tokens bewegt (siehe Abbildung 3.6). Die Hinterwand kommt im geöffneten Zustand auf der Oberfläche des Tisches zu liegen, wodurch der auf ihr angebrachte zweite Code für die Kamera sichtbar wird. So kann durch das ansich zur Positionsbestimmung eingesetzte optische Trackingsystem zuverlässig auch den Öffnungs-Zustand der Tokens auf der Oberfläche erkennen.



Abbildung 3.6: Geöffnetes Container Token

#### **Einbettbare Tokens**

#### **Werkzeug Tokens**

### **3.2.3 Input auf der Tischoberfläche**

Semitransparente Oberfläche, Projektion von unten (Umlenkspiegel), Kamera von unten - Überleitung zur Illumination via Interferenz zwischen Beamer und Kamera

#### **3.2.4 Illumination und Umgebungslichtabhängigkeit**

### **3.3 Erfassung der Benutzerinteraktion durch Software**

### **3.4 Interpretation der Rohdaten und Stabilisierung der Erkennungsleistung**



# **4 Visualisierung und Modellierungsunterstützung**

## **4.1 Technologische Grundlage der Visualisierung**

### **4.1.1 JHotDraw – Überblick**

### **4.1.2 Einsatz von JHotDraw**

### **4.1.3 Projektion von Information auf die Tischoberfläche**

## **4.2 Umsetzung der Anforderungen zur Modellierungsunterstützung**

### **4.2.1 Benennung von Blöcken und Verbindungen**

inkl. Löschen

### **4.2.2 Abstraktion**

Container

### **4.2.3 Modellierungshistorie**

automatisches Tracking vs. Snapshots

### **4.2.4 Wiederherstellungsunterstützung**



# **5 Persistierung**

## **5.1 Möglichkeiten der Persistenzsicherung**

- Serialisierung von Java-Objekten
- Relationale Datenbanken
- XML Topic Maps

## **5.2 Topic Maps**

## **5.3 Abbildung von Modellen auf Topic Maps**

## **5.4 Technische Umsetzung der Persistierung von Modellen**



# 6 Untersuchungsdesign

Die Evaluierung des in der vorliegenden Arbeit beschriebenen Werkzeuges wurde entsprechend der in Kapitel XY vorgestellten konzeptuellen Zusammenhänge mit Fokus auf unterschiedliche Gesichtspunkte durchgeführt. Die beiden grundlegenden Untersuchungsfragen sind

- Unterstützen Werkzeug und Methode Articulation Work?
- Ermöglichen und unterstützen die Teilwerkzeuge des Modellierungstisches Articulation Work?

Die erste Frage setzt im oberen Bereich der Argumentationskette an (**Bild einfügen**) und untersucht, ob Articulation Work bei Einsatz des Werkzeugs tatsächlich auftritt bzw. ob diese unterstützt wird. Die zweite Frage geht wiederum von der Unterstützung von Articulation Work aus, betrachtet hierbei jedoch den Beitrag der vorhandenen Teilwerkzeuge und zielt auf die Untersuchung der Übereinstimmung zwischen intendierter (bzw. aus den Anforderungen ableitbaren) und tatsächlicher Einsatzgebiete ab. Die diesen Fragen zugrunde liegenden Annahmen und die jeweiligen Ansätze zur Messung werden in den nächsten beiden Abschnitten behandelt. Der letzte Abschnitt dieses Kapitels beschreibt dann die konkrete Planung der Untersuchung und die im Einzelnen durchgeführten Untersuchungs-Phasen.

## 6.1 Frage I – Unterstützung von Articulation Work

Axiom 1: Erfolgreiche Articulation Work zeigt sich an der Production Work (-> Ref. Strauss)

Axiom 1,5: Erfolgreiche Production Work zeigt sich an der Zielerreichung (-> Erfolg) (-> Ref. Fujimura)

Axiom 2: (Geschäfts-)Erfolg steht in direktem Zusammenhang mit funktionierender Interaktion (-> Ref.

Messung: Die Unterstützung der Articulation Work kann an Ihren Auswirkungen auf die Production Work gemessen werden, dort im speziellen an der Qualität der

Interaktion ("Work rests ultimately on Interaction"). Messpunkte können dabei die Akteure oder die Ergebnisse der kollaborativen Arbeit sein.

Misst man an den Akteuren, so kann die subjektive Zufriedenheit mit dem Arbeitsprozess (Verlauf der Interaktion) und/oder das beobachtbare Verhalten der Akteure als Merkmal herangezogen werden. Ersteres kann methodisch durch qualitative Interviews (ggf. unterstützt durch Modelle -> Herrmann, Jahnke 2008) beurteilt werden. Zweiteres kann durch Techniken der Interaktionsanalyse bewertet werden, wobei insbesondere die Gegenüberstellung der tatsächlichen zu den vereinbarten Interaktionsmodalitäten und die Veränderung dieser Vereinbarung über die Zeit von Interesse ist. Dazu ist eine Externalisierung der Vereinbarungen zu unterschiedlichen Zeitpunkten notwendig.

Misst man an den Ergebnissen der kollaborativen Arbeit, so kann die subjektive Zufriedenheit mit dem Ergebnis und die "verobjektivierte" (Experten)-Beurteilung der Qualität des Ergebnisses als Merkmal verwendet werden. Ersters wird wiederum qualitativ in Befragungen zu erheben sein. Die Qualität des Ergebnisses wird von Experten an noch zu definierenden Merkmalen gemessen werden, an denen sich die Interaktion bei der Erstellung zeigt (etwa: Stilbrüche, etc.). Bei der Beurteilung der Qualität der Ergebnisse ist vor allem die Gegenüberstellung zu Ergebnissen von Interesse, die ohne Unterstützung der Articulation Work entstanden sind. Dementsprechend kann der Einsatz einer Kontrollgruppe sinnvoll sein.

## 6.2 Frage 2 – Beitrag und Verwendung der Teilwerkzeuge

Axiom: Articulation Work kann durch Strukturlegetechniken unterstützt werden

Messung: Die Verwendung der einzelnen Werkzeuge kann durch Beobachtung und Befragung modellierender Personen sowie der Untersuchung der Modellierungsresultate beurteilt werden. Messpunkte sind damit wiederum die Aktionen und die Ergebnisse der Artikulation. Bei der Messung ist in diesem Zusammenhang kein kollaboratives Setting notwendig, da nicht die Auswirkungen des Werkzeugs auf Interaktion sondern die Verwendung des Werkzeugs selbst untersucht werden soll. Die Modellierung wird also von einzelnen Personen durchgeführt (Selbst-Artikulation, Externalisierung eigneter mentaler Modelle). Durch Auswertung der Modellierungen aus Evaluierung I lässt sich ein etwaiger Unterschied in der Verwendung der Werkzeuge bei kollaborativen Settings belegen.

## 6.3 Untersuchungsablauf

Die Evaluierung selbst wurde in drei Phasen gegliedert, deren Untersuchungsfokus auf

- der Verwendbarkeit des Werkzeugs an sich,
- dessen Eignung zur Unterstützung von "Articulation Work" in Lehr- und Lern-Szenarien und
- dessen Eignung zur Unterstützung von "Articulation Work" beim Einsatz in Unternehmen

lag. Die erste Phase ist als Vorlauf zu sehen, der nicht unmittelbar der Beantwortung der Untersuchungsfragen diente, sondern auf die rein explorative Untersuchung der tatsächlichen Verwendbarkeit des Werkzeuges (im Sinne der Verständlichkeit und Robustheit) abzielte. Die beiden folgenden Phasen decken die Bearbeitung der eigentlichen Untersuchungsfragen ab, wobei durch den unterschiedlichen Einsatzkontext versucht wurde die verschiedenen Anwendungsbereiche des Werkzeugs in die Untersuchung einfließen zu lassen.

Im Zuge der Evaluierungen wurde das Werkzeug unter realen Einsatzbedingungen getestet. Unter "real" ist hier zu verstehen, dass die testenden Benutzer mit der Bedienung des Werkzeugs vorab nicht vertraut waren und dass die Aufgabenstellung stets einen konkreten Bezug zu einem für die jeweiligen Personen relevanten Arbeitsszenario aufwies. In den folgenden Abschnitten wird im Detail auf die Konzeption der einzelnen Phasen und den jeweiligen Beitrag zur Beantwortung der Untersuchungsfragen eingegangen.

### 6.3.1 Phase I – Verwendbarkeit

### 6.3.2 Phase 2 – Lehr- und Lern-Szenarien

### 6.3.3 Phase 3 – Unternehmenseinsatz



# **7 Untersuchungsergebnisse**

## **7.1 Erhobene Daten**

### **7.1.1 Phase 1**

In Phase 1 wurden 9 Modellierungsdurchgänge mit insgesamt 18 Personen durchgeführt. An dem vorangegangenen Pretest nahmen 12 Personen teil.

### **7.1.2 Phase 2**

In Phase 2 wurden Untersuchungen im Rahmen zweier Lehrveranstaltungen durchgeführt. An der ersten Untersuchung nahmen 18 Studierende der Wirtschaftsinformatik teil, die in Gruppen zu 2 Personen insgesamt 17 Modellierungsdurchgänge durchführten. An der zweiten Untersuchung nahmen 54 Studierende in Gruppen zu 3 Personen an insgesamt 18 Modellierungsdurchgängen teil.

### **7.1.3 Phase 3**

## **7.2 Auswertung & Interpretation**



# Literaturverzeichnis

- Beer, W., Christian, V., Ferscha, A., and Mehrmann, L. (2003). Modeling Context-aware Behavior by Interpreted ECA Rules. In *Proceedings of the International Conference on Parallel and Distributed Computing (EUROPAR '03)*, volume 2790 of *LNCS*, pages 1064–1073. Springer.
- Dey, A. K., Salber, D., and Abowd, G. D. (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction (HCI) Journal*, 16(2-4):97–166.
- Fitzmaurice, G. (1996). *Graspable User Interfaces*. Phd-thesis, University of Toronto.
- Fitzmaurice, G., Ishii, H., and Buxton, W. (1995). Bricks: laying the foundations for graspable user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI)*, pages 442–449. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA.
- Fjeld, M. (2001). *Designing for tangible interaction*. PhD thesis, Swiss Federal Institute of Technology.
- Fjuk, A., Nurminen, M., and Smørdal, O. (1997). Taking Articulation Work Seriously: An Activity Theoretical Approach. Technical Report TUCS TR 120, Turku Centre for Computer Science.
- Fujimura, J. (1987). Constructing 'Do-Able' Problems in Cancer Research: Articulating Alignment. *Social Studies of Science*, 17(2):257–293.
- Furtmüller, F. (2007). Implementierung eines Frameworks für berührbare Benutzungsschnittstellen. Master's thesis, University of Linz.
- Furtmüller, F. and Oppl, S. (2007). A Tuple-Space based Middleware for Collaborative Tangible User Interfaces. In *Proceedings of WETICE '07*. IEEE Press.
- Gerson, E. and Star, S. (1986). Analyzing due process in the workplace. *ACM Transactions on Information Systems (TOIS)*, 4(3):257–270.

## Literaturverzeichnis

---

- Hampson, I. and Junor, A. (2005). Invisible work, invisible skills: interactive customer service as articulation work. *New Technology, Work Employment*, 20(2):166 – 181.
- Hanke, U. (2006). *Externe Modellbildung als Hilfe bei der Informationsverarbeitung und beim Lernen*. PhD thesis, University of Freiburg.
- Hughes, F. (1971). *The Sociological Eye*. Aldine de Gruyter.
- Ishii, H. and Ullmer, B. (1997). Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI)*, pages 234–241. ACM Press New York, NY, USA.
- Nonaka, I. and Takeuchi, H. (1995). *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation*. Oxford University Press.
- Oppl, S. (2004). Context-aware Group-Interaction. Master's thesis, University of Linz, Department for Pervasive Computing.
- Patten, J., Ishii, H., Hines, J., and Pangaro, G. (2001). Sensetable: a wireless object tracking platform for tangible interfaces. In *Proceedings of the ACM Conference on Human Factors in Computing Systems 2001 (CHI '01)*.
- Rumelhart, D. and Norman, D. (1978). Accretion, tuning, and restructuring: Three modes of learning. In Cotton, J. and Klatzky, R., editors, *Semantic factors in cognition*, pages 37–53. Erlbaum, Hillsdale, N.J.
- Schmidt, K. (1990). Analysis of Cooperative Work. A Conceptual Framework. Technical Report Risø-M-2890, Risø National Laboratory.
- Seel, N. M. (1991). *Weltwissen und mentale Modelle*. Hogrefe, Göttingen u.a.
- Semmer, N. and Udris, I. (2004). Bedeutung und Wirkung von Arbeit. In Schuler, H., editor, *Lehrbuch Organisationspsychologie*, pages 157–195. Huber, Bern, 3rd edition.
- Strauss, A. (1985). Work and the Division of Labor. *The Sociological Quarterly*, 26(1):1–19.
- Strauss, A. (1988). The Articulation of Project Work: An Organizational Process. *The Sociological Quarterly*, 29(2):163–178.
- Strauss, A. (1993). *Continual Permutations of Action*. Aldine de Gruyter, New York.

# **Index**

- Barcode, 14
- Bluetooth, 20
- Context Toolkit, 22
- Positionsbestimmung, 14
  - akustisch, 17
  - elektromagnetisch, 16
  - kapazitiv, 15
  - optisch, 14
- RFID, 16
- SmartIT, 19
- Token
  - aktive, 19
  - passive, 18
- Ultraschall, 17
- ZigBee, 20