



hochschule mannheim

# **Ein formales Modell zur Beschreibung geräteübergreifender Interaktionsmuster**

Horst Schneider

Master-Thesis

zur Erlangung des akademischen Grades Master of Science (M.Sc.)

Studiengang Informatik

Fakultät für Informatik

Hochschule Mannheim

27.11.2015

Betreuer

Prof. Thomas Smits, Hochschule Mannheim

Prof. Kirstin Kohler, Hochschule Mannheim

**Schneider, Horst:**

Ein formales Modell zur Beschreibung geräteübergreifender Interaktionsmuster / Horst Schneider. –

Master-Thesis, Mannheim : Hochschule Mannheim, 2015. 93 Seiten.

**Schneider, Horst:**

A formal model for the description of interaction patterns between devices / Horst Schneider. –

Master-Thesis, Mannheim : University of Applied Sciences Mannheim, 2015. 93 pages.

## **Erklärung**

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ich bin damit einverstanden, dass meine Arbeit veröffentlicht wird, d. h. dass die Arbeit elektronisch gespeichert, in andere Formate konvertiert, auf den Servern der Hochschule Mannheim öffentlich zugänglich gemacht und über das Internet verbreitet werden darf.

Mannheim, 27.11.2015

Horst Schneider



# Abstract

## ***Ein formales Modell zur Beschreibung geräteübergreifender Interaktionsmuster***

Um Designern und Entwicklern zu ermöglichen, intuitiv nutzbare Übergänge zwischen verschiedenen Geräten in einem zusammenhängenden Nutzungskontext zu gestalten, wurde im Projekt SysPlace ein Katalog von Entwurfsmustern entwickelt. Bisher fehlt in diesen Entwurfsmustern allerdings eine einheitliche Möglichkeit, die technische Umsetzung der Interaktionen konsistent und formal zu beschreiben.

In der vorliegenden Arbeit wird mittels einer vergleichenden Literaturrecherche eine Abstraktion der technischen Aspekte erarbeitet, die geräteübergreifende Interaktionen implementieren. Das Ergebnis ist ein Modell, das eine formale Beschreibung der erarbeiteten Abstraktionen darstellt und in die Entwurfsmuster integriert werden kann. Anhand ausgewählter Entwurfsmuster wird die Verwendung des Modells veranschaulicht.

## ***A formal model for the description of interaction patterns between devices***

The SysPlace project developed a design pattern catalogue for intuitive transitions between devices, which allows designers and software developers to create a coherent user experience. The patterns lack a way to describe the interaction's technical implementation uniformly and consistently.

In the scope of this thesis, we conducted a literature research in order to create an abstraction including the different technical aspects of interactions between devices. The result is a model, which allows a formal description of the suggested abstractions and can be integrated into existing design patterns. We illustrate the use of this model with selected design patterns.



# Inhaltsverzeichnis

<b>1. Einleitung</b>	<b>1</b>
1.1. Das Projekt SysPlace . . . . .	1
1.2. Motivation und Zielsetzung . . . . .	2
1.3. Methodik . . . . .	3
1.4. Aufbau . . . . .	3
<b>2. Technische und konzeptionelle Grundlagen</b>	<b>5</b>
2.1. Multiscreen . . . . .	5
2.2. Geräteklassen für Multiscreen-Interaktionen . . . . .	7
2.3. Multiscreen-Patterns . . . . .	10
2.4. Relevante Technologien . . . . .	12
2.4.1. Konnektivität . . . . .	12
2.4.2. Sensoren . . . . .	18
2.4.3. Ausgabe . . . . .	23
<b>3. Modell für Multiscreen-Interaktionen</b>	<b>25</b>
3.1. Herausforderungen . . . . .	25
3.2. Anforderungen an das Modell . . . . .	26
3.3. Erarbeitung des Modells . . . . .	29
3.3.1. Rahmenmodell . . . . .	30
3.3.2. Erkennung einfacher Gesten . . . . .	34
3.3.3. Erkennung synchroner Gesten . . . . .	39
3.3.4. Connect . . . . .	45
3.3.5. Select . . . . .	51
3.3.6. Transfer . . . . .	55
3.3.7. Disconnect . . . . .	60
3.3.8. Statische Sicht – Domänenmodell . . . . .	62
<b>4. Validierung des Modells</b>	<b>69</b>
4.1. Qualitative Bewertung der Anforderungserfüllung . . . . .	69
4.2. Exemplarische Instanziierungen des Modells . . . . .	73
4.2.1. Bump-Patterns . . . . .	73
4.2.2. Approach-Patterns . . . . .	81
4.3. Diskussion der Anwendbarkeit für weitere Patterns . . . . .	87

4.4. Ansatz zur Entwicklung eines Frameworks . . . . .	89
<b>5. Zusammenfassung und Ausblick</b>	<b>91</b>
5.1. Zusammenfassung der Ergebnisse . . . . .	91
5.2. Weiterführende Arbeiten . . . . .	92
<b>Abkürzungsverzeichnis</b>	<b>vii</b>
<b>Tabellenverzeichnis</b>	<b>ix</b>
<b>Abbildungsverzeichnis</b>	<b>xi</b>
<b>Literaturverzeichnis</b>	<b>xv</b>
<b>Index</b>	<b>xxi</b>
<b>A. Patterntemplate SysPlace</b>	<b>xxi</b>
<b>B. Übersicht SysPlace-Patterns</b>	<b>xxxiii</b>
<b>C. Abwandlungen der synchronen Gestenerkennung</b>	<b>xxxix</b>



# Kapitel 1

## Einleitung

### 1.1. Das Projekt SysPlace

Diese Master-Thesis wird im Rahmen des Projektes SysPlace (BMBF-gefördert, Förderkennzeichen 01IS14018D) verfasst. Das allgemeine Projektziel ist es, softwareentwickelnde kleine und mittelständische Unternehmen (KMU) in die Lage zu versetzen, geräteübergreifende Systeme zu entwickeln, die neben einem bestmöglichen Nutzererlebnis eine hohe Kosten- / Nutzen Relation für KMU ermöglichen. Hierfür sollen systematische Anleitungen zur Interaktionsgestaltung und Implementierung in Form von Entwurfs- und Entwicklungsansätzen, Guidelines und wiederverwendbaren Software-Komponenten entwickelt und validiert werden.

Drei zentrale Forschungsfragen wurden für das Projekt formuliert:

1. Wie lassen sich die Stärken der natürlichen Interaktion durch die Konzeption geeigneter Gesten auf das Gebiet der Ecosystems of Displays übertragen?
2. Welche methodisch-technischen Voraussetzungen sind notwendig, um diese Interaktionen zu unterstützen?
3. Welche Entwicklungsverfahren, -werkzeuge und -modelle und Voraussetzungen muss die zugrunde liegende Softwarearchitektur des Gesamtsystems mitbringen, um die Szenarien zu unterstützen?

Die Erkenntnisse dieser Arbeit sollen vor allem einen Beitrag zur Beantwortung der letzten beiden Fragen liefern und basieren auf bereits erarbeiteten Forschungsergebnissen, die im folgenden Kapitel noch näher erläutert werden.

### 1.2. Motivation und Zielsetzung

Die meisten Menschen verbringen einen Großteil ihres Tages vor Bildschirmen verschiedenster Art, von Smartphones über Tablets zu Desktop-Rechnern und internet-fähigen Fernsehern. Verschiedene Interaktionskonzepte sind entstanden, um diese Geräteübergänge für den Anwender intuitiv nutzbar zu machen. Aufgrund der heterogenen Darstellung und Umsetzung von Multiscreen-Konzepten werden diese im Forschungsprojekt SysPlace strukturiert und kategorisiert in Form von Interaktionsmustern erfasst und Designern sowie Entwicklern zur Verfügung gestellt. Aufgrund der hohen technischen Komplexität dieser Interaktionskonzepte fehlt bisher die Möglichkeit, die Details zur technischen Realisierung strukturiert und einheitlich in die Beschreibungen der Interaktionsmuster aufzunehmen.

Ziel dieser Arbeit ist es, die technischen Aspekte der verschiedenen Interaktionsmuster zu analysieren und eine formale Basis zu entwickeln, um diese einheitlich, abstrakt und plattformunabhängig beschreiben zu können. Dabei liegt der besondere Fokus auf der flexiblen Gestaltungsmöglichkeit von Interaktionsdetails.

Einzelne Interaktionskonzepte wurden hinsichtlich ihrer technischen Machbarkeit bereits von verschiedenen Institutionen und Forschern intensiv wissenschaftlich untersucht und prototypisch umgesetzt, zumeist mit einer an eine bestimmte Technologie gekoppelten Lösung. Zusammen mit den bereits im Projekt SysPlace konzipierten Prototypen bilden diese Vorarbeiten eine breite Basis, um daraus allgemeine Regeln für die Implementierung von Interaktionen im Multiscreen-Kontext abzuleiten.

Das Ergebnis der Arbeit ist ein Modell, das eine abstrakte, einheitliche Beschreibung der technischen Realisierung verschiedener Interaktionsmuster ermöglicht. Es soll Entwicklern die Möglichkeit bieten, die erforderlichen Komponenten, Abläufe und Nutzerinteraktionen sowie deren semantischen Zusammenhänge zu verstehen und konkret umsetzen zu können. Um das zu erreichen, wird die Modellbeschreibung so gestaltet, dass sie sich einfach in ausführbare Komponenten übersetzen lässt.

Im Fokus der Arbeit steht dabei nicht, ein fertiges Framework zu entwickeln oder einzelne Interaktionsmuster mit konkreten Technologien zu realisieren. Zudem wird keine Bewertung vorgenommen, inwieweit sich einzelne Technologien zur Umsetzung bestimmter Aspekte von Interaktionsmustern eignen.

### **1.3. Methodik**

Das Thema wird theoretisch auf Basis einer vergleichenden Literaturrecherche bearbeitet. Es werden Publikationen betrachtet, in denen geräteübergreifende Interaktionen theoretisch entworfen und anhand prototypischer Systeme evaluiert wurden. Den Ausgangspunkt bildet die Literatur, die bereits bei der Erfassung der Patterns im Projekt SysPlace betrachtet wurde. Es werden allerdings auch weitere Publikationen in die Recherche einbezogen, die den gleichen Forschungsgegenstand haben.

Das Ziel ist es, Gemeinsamkeiten und Unterschiede zwischen den verschiedenen Ansätzen zu analysieren und eine Abstraktion herauszuarbeiten, mit denen die heterogenen Konzepte geräteübergreifender Interaktionen einheitlich erfasst werden können. Das Ergebnis soll ein Modell sein, mit dem die technischen Realisierungsmöglichkeiten der analysierten Systeme sowie weiterer Systeme im Multiscreen-Kontext einheitlich beschrieben werden können.

Zur Validierung der Ergebnisse werden einige Patterns anschließend exemplarisch durch das Modell beschrieben. Darauf aufbauend wird diskutiert, inwieweit sich die bisher erfassten Patterns durch das Modell beschreiben lassen, bzw. ob eine Erweiterbarkeit um neue Patterns gegeben ist.

### **1.4. Aufbau**

Die Einleitung in diesem Kapitel stellt das Projekt vor, in dessen Rahmen diese Arbeit verfasst wurde und beschreibt anschließend die Motivation und Zielsetzung sowie die Methodik, die zum Erreichen des Ziels angewendet wird. Im zweiten Kapitel werden technische und konzeptionelle Grundlagen geräteübergreifender Interaktion beschrieben, die zum Verständnis der im Hauptkapitel analysierten Literatur relevant sind. Kapitel 3 beinhaltet eine ausführliche Analyse geräteübergreifender Interaktionen und bildet somit den eigenen Beitrag der Arbeit, indem ein Modell erarbeitet wird, das die in der Einleitung vorgestellte Problemstellung löst. Eine Validierung der Ergebnisse wird in Kapitel 4 entsprechend der beschriebenen Methodik vorgenommen. Abschließend wird eine Zusammenfassung der erarbeiteten Ergebnisse sowie ein Überblick über mögliche weiterführende in Kapitel 5 gegeben.



## Kapitel 2

# Technische und konzeptionelle Grundlagen

In diesem Kapitel werden technische Grundlagen und Begriffsdefinitionen behandelt, die für das Verständnis der Arbeit relevant sind. Auf konzeptioneller Ebene werden zunächst zentrale Begriffe und deren Verwendung innerhalb der Arbeit definiert. Anschließend wird ein Überblick über Zweck und Verwendung von Entwurfsmustern bei der Oberflächen- und Interaktionsgestaltung gegeben.

Auf der technischen Ebene wird zunächst das Spektrum vorhandener Gerätetypen für Multiscreen-Szenarien vorgestellt, klassifiziert und eingeschränkt. Anschließend werden Technologien beschrieben, die besonders geeignet für die Realisierung von Multiscreen-Anwendungen sind. Da ein Gesamtüberblick unmöglich ist, wird hier eine Auswahl von verbreiteten und aktuell verfügbaren Technologien vorgenommen.

### 2.1. Multiscreen

Das zentrale Thema dieser Arbeit ist die Gestaltung von Interaktionen, die den Nutzer in die Lage versetzen, eine zusammenhängende Aufgabe geräteübergreifend auf mehreren Displays zu erledigen.

Ein häufig in der Literatur zu findender Begriff ist der des *multi-display environment* (MDE), der von Seyed u. a. wie folgt definiert wird [Seyed u. a., 2012]:

A multi-display environment (MDE) is a system where interaction is divided over several displays, such as digital tabletops, wall displays and personal devices like tablets or mobile phones. MDEs often include

## 2. Technische und konzeptionelle Grundlagen

---

heterogeneous displays to take advantage of different capabilities such as their size, position, resolution or mobility to support the task at hand.

Der Begriff MDE ist in der Fachliteratur seit gut einer Dekade zu finden und wird bereits von Rekimoto [1997] für die Kombination mehrerer Geräte wie Desktop-PCs, Personal Digital Assistants (PDAs) und größeren Wandbildschirmen verwendet.

Um den interaktiven Aspekt bei der gemeinsamen Nutzung verschiedener Geräte noch mehr hervorzuheben, sprechen Terrenghi und Kollegen von einem *ecosystem of displays*, das an den jeweiligen Nutzungskontext angepasst werden kann [Terrenghi u. a., 2009].

Mit *symphony of devices* definieren Hamilton und Kollegen einen weiteren Begriff, der bewusst gewählt wurde, um eine Ad-hoc-Kombination heterogener Geräte, je nach Anwendungsfall, in den Vordergrund zu stellen. Zudem soll dadurch eine Abgrenzung zu anderer Forschung stattfinden, in der dieser Aspekt nicht explizit beleuchtet wird [Hamilton und Wigdor, 2014].

Diese Arbeit folgt einer Definition von Nagel und Fischer. Diese verwenden die Begriffe *device* und *screen* (und damit auch *display*) synonym für digitale Endgeräte [Nagel und Fischer, 2013, 16]. Mittels solcher Endgeräte können Informationen abgerufen, dargestellt und bearbeitet werden. Der Bildschirm eines Gerätes ist ausschlaggebend für die Art und Repräsentation der Informationen, die abgerufen werden können. Zudem ist bei mobilen Endgeräten der Bildschirm zumeist fest verbaut und bildet damit eine Einheit mit dem eigentlichen Gerät, weshalb diese Gleichsetzung der Begriffe *device* und *screen* sinnvoll ist.

Darauf aufbauend wird für die Kombination mehrerer Geräte in einem gemeinsamen Nutzungskontext der Begriff *Multiscreen*, im Singular zumeist die Begriffe Gerät und Endgerät verwendet [Nagel und Fischer, 2013]. Auch bei Google findet sich der Begriff Multiscreen in einer umfangreichen Studie, die die Nutzung verschiedener Geräte wie Smartphones, Tablets und PCs zur Erledigung zusammenhängender Aufgaben analysiert [Google, 2012]. Damit bezeichnet Multiscreen nicht nur einen Rechner mit mehreren Monitoren, sondern die Kombination verschiedener heterogener Endgeräte.

Darüber hinaus werden in dieser Arbeit die folgenden Begriffe verwendet:

- *(End)gerät*: Ein einzelnes Gerät (z. B. Desktop-PC, Tablet etc.), das in einem Multiscreen-Szenario verwendet werden kann.

- *Multiscreen-Szenario*: Ein Anwendungsfall, in dem verschiedene (heterogene) Geräte zur Erledigung einer zusammenhängenden Aufgabe genutzt werden [Nagel und Fischer, 2013, 19].
- *Multiscreen-Anwendung*: Eine Anwendung, die auf verschiedenen Endgeräten nutzbar ist und Daten geräteübergreifend bereitstellen kann [Nagel und Fischer, 2013, 18]. Eine Multiscreen-Anwendung kann ein Multiscreen-Szenario teilweise oder ganz realisieren.
- *Multiscreen-Interaktion*: Interaktionen, die dem Nutzer ermöglichen, geräteübergreifend zu arbeiten (z. B. Verbinden von Geräten, Datenübertragung etc.); Begriffsprägung analog zu Multiscreen-Szenario und -Anwendung.
- *Multiscreen-Pattern*: Ein Entwurfsmuster, das die Gestaltung und Implementierung von Multiscreen-Interaktionen unterstützen soll; begriffliche Einordnung der Patterns in den Multiscreen-Kontext.

Zudem unterscheidet Google zwei Modi für die Multiscreen-Nutzung [Google, 2012]:

- *Sequentielle Nutzung*: der Nutzer wechselt während einer zusammenhängenden Aktivität von einem Gerät zum anderen.
- *Simultane Nutzung*: der Nutzer verwendet zwei Geräte parallel für zusammenhängende oder unzusammenhängende Aktivitäten.

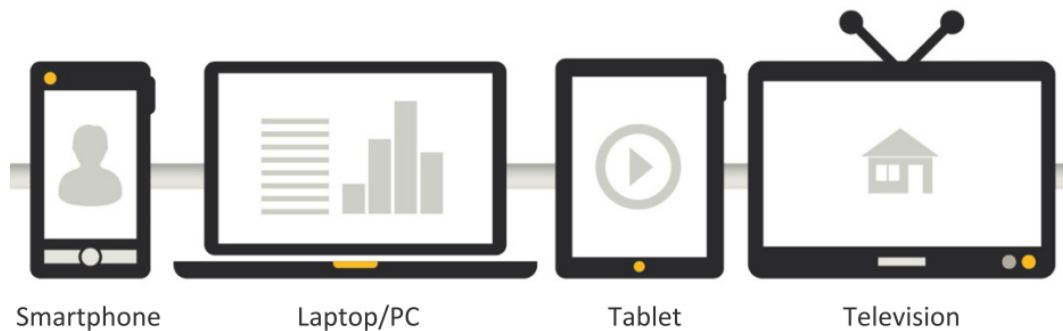
Im folgenden Kapitel werden Geräte, die für Multiscreen-Szenarien in Frage kommen, näher erläutert und in Klassen eingeteilt, sodass sie im weiteren Verlauf der Arbeit eindeutig referenziert werden können.

## 2.2. Geräteklassen für Multiscreen-Interaktionen

Es existiert eine kaum überschaubare Anzahl verschiedenster Geräte, die in Multiscreen-Szenarien einsetzbar sind. Dabei stehen klassische Desktop-PCs bzw. Laptops nicht mehr im Mittelpunkt, sondern machen nur noch etwa ein Viertel der täglichen Medieninteraktion aus und wurden bei der Nutzungshäufigkeit von Smartphones verdrängt [Google, 2012]. Schon lange bevor Smartphones allgegenwärtig wurden, gab es mobile Geräte unter verschiedenen Bezeichnungen wie PDA, PocketPC, Palm etc., deren Einsatzmöglichkeiten in Multiscreen-Szenarien bereits früh erforscht wurden [Hinckley u. a., 2004].

## 2. Technische und konzeptionelle Grundlagen

---



**Abbildung 2.1.:** Die vier Geräteklassen Smartphone, PC/Laptop, Tablet, (Smart) TV. Quelle: [Google, 2012].

Apple konnte 2010 mit der Einführung des iPad, das sich in weniger als drei Monaten mehr als 3 Millionen mal verkaufte, Tablets als zusätzliche Geräteklasse auf dem Markt etablieren [Apple, 2010]. Tablets werden heute von vielen verschiedenen Herstellern angeboten und vorwiegend zur Unterhaltung und Kommunikation eingesetzt [Google, 2012]. Die Bildschirmgröße und technische Ausstattung von Tablets können stark variieren.

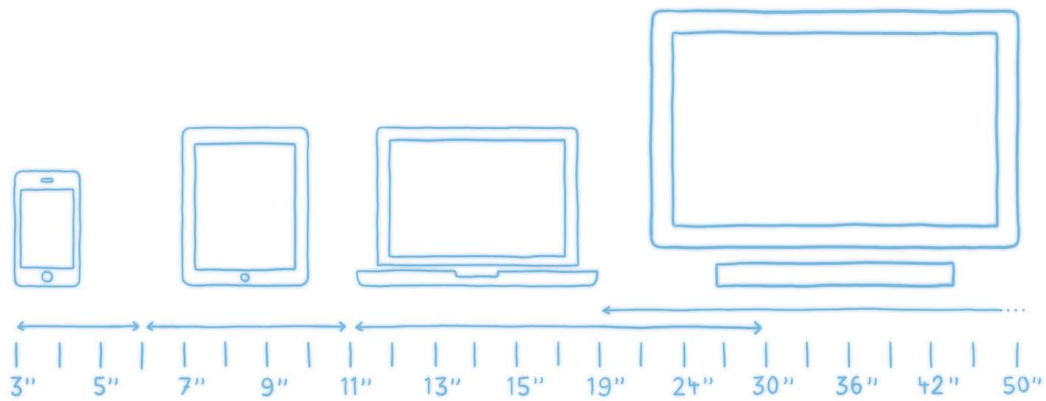
Das klassische TV ist nach wie vor präsent, wird aber meist in Verbindung mit einem anderen Gerät simultan benutzt [Google, 2012] und ist eher ein „Hintergrundrauschen“ [Microsoft, 2013].

Google unterscheidet in seiner Studie zwischen vier verschiedenen Geräteklassen sowie üblichen Nutzungsszenarien und -orten [Google, 2012]:

1. *Smartphone*: Zu Hause oder unterwegs für Kommunikation und schnelle Informationsbeschaffung.
2. *Tablet*: Überwiegend zur Unterhaltung zu Hause.
3. *PC/Laptop*: Produktives, konzentriertes Arbeiten zu Hause oder im Büro.
4. *TV*: Zu Hause zur Unterhaltung, oft parallel mit Geräten anderer Klassen.

Microsoft hat dieselben vier Kategorien und sehr ähnliche Nutzungskontexte herausgearbeitet [Microsoft, 2013]. Auch bei Nagel und Fischer werden diese Kategorien unter der Bezeichnung „Vier Screens“ genannt, eine interessante Ergänzung im Hinblick auf Multiscreen-Interaktion ist hier allerdings die Einschränkung der Kategorie TV auf internetfähige Geräte, sog. *Smart TVs* [Nagel und Fischer, 2013, 31]. Erst durch die Vernetzung mit anderen Geräten können TV-Geräte in Multiscreen-Szenarien wirklich sinnvoll verwendet werden.





**Abbildung 2.2.:** Die vier Geräteklassen und deren übliche Screengröße in Relation. Quelle: [Nagel und Fischer, 2013].

Zwei Ergänzungen zu diesen vier Kategorien werden für diese Arbeit vorgenommen. Auf der einen Seite sind für Multiscreen-Szenarien auch größere Displays als Smart TVs denkbar. So ermöglicht z. B. das System „Dynamo“ das Teilen von Daten über ein öffentlich nutzbare Display, das eine gesamte Wand einnimmt [Izadi u. a., 2003]. Um solche Displays für Multiscreen-Anwendungen nutzbar zu machen, müssen sie netzwerkfähig sein, weshalb sie einfach zu den Smart TVs gezählt werden können.

Auf der anderen Seite stehen Geräte wie Fitnesstracker, Datenbrillen oder mit Sensoren versehene Kleidungsstücke, die direkt am Körper getragen werden. Diese Geräte werden oft als *Wearables* bezeichnet. Im Rahmen dieser Arbeit wird diese Kategorie ausgeklammert, da die Interaktion mit solchen Geräten entweder sehr komplex ist, wie z. B. bei Augmented Reality (AR)- und Virtual Reality (VR)-Brillen, oder aufgrund weniger Sensoren kaum explizit möglich ist, wie z. B. Yoon [2014] feststellt.

Lediglich sog. *Smartwatches*, die seit dem Erscheinen der Pebble im Jahr 2012 als alleinstehende oder ergänzende Geräte zu Tablets oder Smartphones große Verbreitung finden<sup>1</sup>, könnten in Multiscreen-Szenarien zum Einsatz kommen. Die Voraussetzung ist, dass sie programmierbar sind und dem Anwender explizite Interaktionsmöglichkeiten bieten. Da die Anwendungsfälle bzgl. Multiscreen-Interaktionen denen von Smartphones ähneln, werden Smartwatches in dieser Arbeit nicht gesondert betrachtet. Für Smartphones, Tablets und Smartwatches wird der Sammelbegriff *mobile Geräte* verwendet.

<sup>1</sup>Prognosen gehen davon aus, dass der Markt für Smartwatches bis 2018 auf 214 Millionen Stück wachsen wird [Rawassizadeh u. a., 2014].

Weil diese Arbeit auf einer Sammlung von Entwurfsmustern zur Multiscreen-Interaktion aufbaut, wird im nächsten Kapitel der Begriff des Patterns definiert und in Bezug zu Multiscreen-Interaktionen gesetzt.

### 2.3. Multiscreen-Patterns

Entwurfsmuster (engl. *design patterns*) haben ihren Ursprung in der Architektur und werden durch das folgende, obligatorische Zitat von Alexander präzise und allgemeingültig definiert [Alexander u. a., 1977]:

Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.

Auch – oder gerade besonders – in der Informatik gibt es viele wiederkehrende Probleme, für deren Lösung Patterns hilfreich sein können. Die erste und wohl bekannteste Patternsammlung, die im Bereich der Informatik von Gamma und Kollegen veröffentlicht wurde, behandelt eine Vielzahl von Problemstellungen der objektorientierten Programmierung [Gamma u. a., 1995]. Speziell im Bereich der GUI- und Interaktionsgestaltung gibt es verschiedene Patternsammlungen, wobei die von Yahoo [2015], Tidwell [2010] und van Welie [2008] zu den bekanntesten gehören. Während Yahoo überwiegend typische Webdesignprobleme wie Tabs, Fortschrittsbalken und Autovervollständigung adressiert, finden sich bei Tidwell und van Welie z. B. auch Patterns zum Designkontext (Nutzererwartungen an die Webseite, verschiedene Seitentypen etc.) oder der Visualisierung von Daten.

Alle Patternsammlungen haben die Gemeinsamkeit, dass die Patterns in einer jeweils konsistenten, strukturierten Form erfasst werden. Das hat den Vorteil, dass Patterns einfacher zu lernen, zu vergleichen und anzuwenden sind [Gamma u. a., 1995, 6]. Die konkrete Struktur variiert zwischen den einzelnen Patternsammlungen, folgende Bestandteile sind aber meist vorhanden:

- *Eindeutiger Name*: Wie lässt sich das Pattern prägnant nennen? Hilfreich für die eindeutige Referenzierung von Patterns.
- *Problembeschreibung*: Welches Problem löst das Pattern?

- *Lösung*: Welche Schritte sind notwendig, um das Problem zu lösen? Je nach Domäne des Patterns ergänzt um konkrete Lösungshinweise, wie z. B. Code-Beispiele.
- *Wann*: In welchem Kontext kann das Pattern eingesetzt werden, in welchem eher nicht?
- *Warum*: Aus welchen Gründen ist das Pattern zur Lösung des Problems geeignet?
- *Bekannte Anwendungen*: Wo wurde das Pattern bereits erfolgreich eingesetzt?

Die Patternsammlung, die im Projekt SysPlace erarbeitet wird, folgt einer analogen Struktur. Jedes Pattern wird mittels eines *Templates* erfasst. Ein ausgefülltes Template ist exemplarisch in Anhang A zu finden. Da Multiscreen-Interaktionen mit verschiedenen Gerätekombination und in vielfältigen Anwendungskontexten stattfinden können, wird in den Punkten „Wann“ (geeignete Nutzungskontexte) und „Warum“ (Begründung der Intuitivität der Interaktion) dargelegt, warum das Pattern in bestimmten Situationen eine geeignete Multiscreen-Interaktion realisiert. Zur weiteren Strukturierung ist die Patternsammlung in fünf Kategorien unterteilt:

- *Give*: Daten an ein anderes Gerät senden.
- *Take*: Daten von einem anderen Gerät empfangen.
- *Exchange*: Dateien zwischen zwei Geräten austauschen / synchronisieren.
- *Extend*: Mehrere Bildschirme koppeln.
- *Connect*: Zwei Geräte miteinander verbinden.

Der Name jedes Patterns folgt dem Namensschema *<Verb> To <Kategorie>*, also z. B. *Bump To Exchange* oder *Swipe To Give*. Das Verb beschreibt möglichst prägnant die zugrunde liegende *Geste*<sup>2</sup> des Patterns, die Kategorie definiert die Art der angestoßenen Systemreaktion. Das Pattern „Swipe To Give“ bezeichnet demnach eine Wisch-Geste des Nutzers, die die Übertragung einer Datei von einem Gerät auf ein anderes anstößt. In Anhang B ist eine Zusammenfassung aller Patterns zu finden.

---

<sup>2</sup>Der Begriff Geste wird hier analog zu Echtler im weitest möglichen Sinne gebraucht, also als Reaktion eines Systems auf eine Untermenge aller möglichen Nutzereingaben (Multitouch, Handbewegungen, Eingaben mit einem digitalen Stift etc.) [Echtler und Butz, 2012].

Die zentrale Problemstellung, die in dieser Arbeit bearbeitet wird, ist innerhalb des Templates unter dem Punkt „Technisches“ bei der Beschreibung des Lösungsansatzes verortet, bezieht aber auch weitere Aspekte mit ein.

### 2.4. Relevante Technologien

Multiscreen-Anwendungen realisieren Interaktionskonzepte, die vielfältige Anforderungen an die Hard- und Software der verwendeten Geräte stellen. Neben der Konnektivität spielen sowohl Sensoren (zur Messung von Geräteeingaben verschiedener Art) als auch Ausgabetechnologien eine entscheidende Rolle. In den folgenden Unterkapiteln werden die wichtigsten Technologien vorgestellt und deren Einsatzmöglichkeiten aufgezeigt.

#### 2.4.1. Konnektivität

##### *Ethernet und WLAN*

Die am weitesten verbreitete Technologie zum Verbinden von Geräten ist Ethernet<sup>3</sup>, wobei heute zumeist der Protokollstandard IEEE 802.3 oder Abwandlungen davon eingesetzt werden. Ethernet ist kabelgebunden und erreicht in heute gängigen Standards üblicherweise eine Übertragungsgeschwindigkeit von 100 MBit/s bis 10 GBit/s auf Distanzen von 100 Metern (Kupferkabel) bis zu mehreren Kilometern (Glasfaserkabel) [Kurose und Ross, 2014, 506-507]. In Multiscreen-Szenarien ist Ethernet allerdings nur eingeschränkt für Geräte der Kategorie PC/Laptop oder Smart TV einsetzbar, aber nicht für mobile Geräte.

Unter dem Begriff Wireless Local Area Network (WLAN)<sup>4</sup> haben sich Funknetzwerke als Alternative zum kabelgebundenen Ethernet durchgesetzt. Obwohl der Begriff WLAN für viele Arten von Funknetzwerken verwendet wird, ist damit meist ein Standard der Normenfamilie IEEE 802.11 gemeint. Die Übertragungsgeschwindigkeit ist oft niedriger als bei Ethernet-Netzwerken, wobei neuere Standards inzwischen auch in den GBit-Bereich vordringen. Mit den gängigen Standards (vgl. Tabelle 2.1) lassen sich Distanzen von ca. 10 – 30 m überbrücken [Kurose und Ross, 2014, 552].

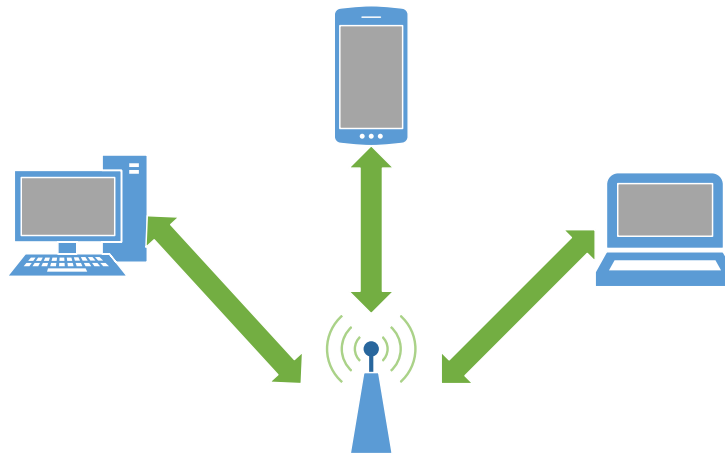
---

<sup>3</sup>Oft synonym verwendet mit Local Area Network (LAN) oder Ethernet-LAN.

<sup>4</sup>Oft synonym mit Wi-Fi verwendet.

Standard	Frequenzbereich (USA)	Übertragungsrate
802.11b	2.4 – 2.485 GHz	bis 11 MBit/s
802.11a	5.1 – 5.8 GHz	bis 54 MBit/s
802.11g	2.4 – 2.485 GHz	bis 54 MBit/s

**Tabelle 2.1.:** Gängige WLAN-Standards. Quelle: [Kurose und Ross, 2014, 562].



**Abbildung 2.3.:** WLAN im Infrastrukturmodus.

Drahtlose Netzwerke können in zwei Modi betrieben werden [Kurose und Ross, 2014, 555]:

- *Infrastruktur-Modus:* Alle Geräte im Netzwerk kommunizieren mit einer Basisstation, die wiederum mit weiteren Netzwerken oder dem Internet verbunden sein kann (siehe Abbildung 2.3).
- *Ad-hoc-Modus:* Es gibt keine Basisstation, alle Geräte kommunizieren direkt miteinander (siehe Abbildung 2.4).

Für beide Modi gibt es in Multiscreen-Szenarien Anwendungsmöglichkeiten. Die meisten mobilen Geräte sowie PCs und Laptops verfügen über einen WLAN-Adapter. Im Infrastruktur-Modus können Geräte aller Kategorien über die Basisstation miteinander kommunizieren, entweder direkt per WLAN oder durch an die Basisstation angeschlossene Geräte wie z.B. Smart TVs, die nur eine Ethernet-Schnittstelle besitzen. Um ganz auf die Basisstation zu verzichten, können Multiscreen-Anwendungen auch den Ad-hoc-Modus nutzen. Das hat den Vorteil, dass Geräte auch ohne Anmeldung an einer Basisstation miteinander verbunden werden

## 2. Technische und konzeptionelle Grundlagen

---

können. Damit entfällt die Notwendigkeit, für Multiscreen-Anwendungen eine feste Netzwerkinfrastruktur einzurichten.

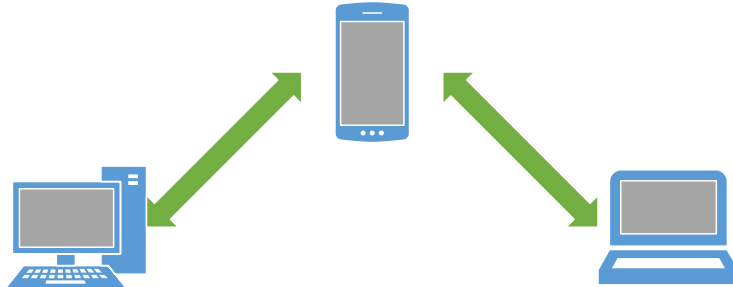


Abbildung 2.4.: WLAN im Ad-Hoc-Modus.

Mit Wi-Fi Direct wird im nächsten Kapitel ein Standard vorgestellt, der es ermöglicht, den Ad-hoc-Modus relativ einfach universell einzusetzen.

### **Wi-Fi Direct**

Wi-Fi Direct ist ein von der Wi-Fi Alliance<sup>5</sup> spezifizierter Standard zur direkten, drahtlosen Verbindung von Geräten basierend auf den IEEE 802.11 Netzwerkstandards. Mit Wi-Fi Direct ist es möglich, zwei Geräte direkt oder mehrere Geräte als Gruppe miteinander zu verbinden, um z. B. Dateien auszutauschen, Geräte zu synchronisieren oder Drucker anzusteuern [Wi-Fi Alliance, 2015b].

Ebenso ermöglicht Wi-Fi Direct, Geräte in der Nähe zu finden und die eigentliche Übertragung von Daten durchzuführen. Eine Basisstation oder Internetverbindung ist nicht nötig, es wird also ein Ad-hoc-Modus realisiert, bei dem ein Wi-Fi Direct fähiges Gerät die Verantwortung für die Gruppenkoordination übernehmen kann [Wi-Fi Alliance, 2015b].

Für den Einsatz in Multiscreen-Anwendungen spricht, dass Wi-Fi Direct herstellerunabhängig ist und über System-APIs auf einer Vielzahl vor allem mobiler Endgeräte einfach verwendet werden kann. Zudem können die hohen Übertragungsgeschwindigkeiten des jeweils zugrunde liegenden Drahtlosstandards genutzt werden, was den Austausch größerer Datenmengen ermöglicht, als es z. B. bei Bluetooth oder NFC der Fall ist.

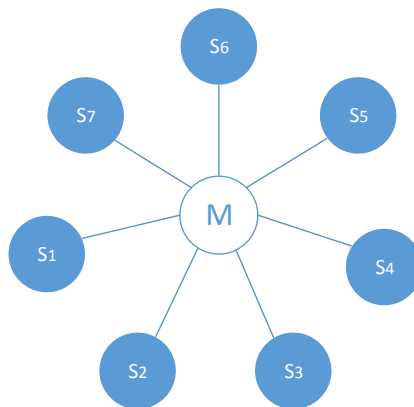
---

<sup>5</sup>Die Wi-Fi Alliance ist eine 1999 gegründete Non-Profit-Organisation mit dem Ziel, herstellerunabhängige Standards für die drahtlose Kommunikation zu spezifizieren und deren Einhaltung zu zertifizieren [Wi-Fi Alliance, 2015a].

### Bluetooth

Bluetooth ist ein weit verbreiteter Standard für eine drahtlose, verbrauchsarme Funktechnologie. Die Spezifikation und Weiterentwicklung des Standards liegt in der Verantwortung der Bluetooth Special Interest Group (SIG). Basierend auf einer 1994 von Ericsson durchgeführten Studie wurde Bluetooth mit dem Ziel entwickelt, drahtlose Datenübertragung im ISM-Band<sup>6</sup> auf kurzen Distanzen zu realisieren [Fuchß, 2009, 121].

Aufgrund der niedrigen Hardwarekosten und dem fast ubiquitären Einsatz von Bluetooth kann eine Vielzahl von Geräten miteinander verbunden werden. Durch das sog. *Pairing* werden zwei Geräte gekoppelt, um einen Verbindungsaufbau zwischen den Geräten zu ermöglichen. Bluetooth folgt dabei immer der Topologie eines *Piconetzes*, in dem ein Gerät (Master) die Koordination von bis zu sieben aktiven Geräten (Slaves) übernimmt [Kurose und Ross, 2014, 579]. Die Kommunikation erfolgt immer über den Master, wodurch sich ein sternförmiges Netz ergibt (siehe Abbildung 2.5).



**Abbildung 2.5.:** Piconetz aus einem Master und der Maximalzahl von sieben Slaves. Angelehnt an [Fuchß, 2009, 125].

Eine wichtige Neuerung in der aktuellen Version 4.2 ist Bluetooth Low Energy (BLE)<sup>7</sup>. Das Bluetooth-Protokoll wurde komplett überarbeitet und der Stromverbrauch deutlich reduziert. Durch BLE können so auch viele Geräte, die vorher aus Gründen der Stromersparnis proprietäre Funkstandards verwendet haben, über den universellen Bluetooth Standard miteinander kommunizieren [Hansen, 2015].

<sup>6</sup>Das Industrial, Scientific and Medical (ISM)-Band ist ein Frequenzbereich (2,4 bis 2,485 GHz), der ohne Lizenzkosten genutzt werden kann und daher universell verwendbar ist.

<sup>7</sup>Auch vermarktet als Bluetooth Smart.

## 2. Technische und konzeptionelle Grundlagen

Der Bluetooth-Standard ist in verschiedenen Versionen verfügbar, in denen jeweils andere Features und Übertragungsbandbreiten spezifiziert sind. Die Versionen 4.0 und höher sind mit den vorigen nicht kompatibel, weshalb oft eine Unterscheidung zwischen BLE und Bluetooth Classic vorgenommen wird, wobei mit Bluetooth Classic meist die Version 2.1 gemeint ist. Bluetooth Classic erreicht Geschwindigkeiten von ca. 700 KBit/s (Version 1.1) bis ca. 2 MBit/s (Version 2.1 + EDR) [Fuchß, 2009, 129].

Die Übertragungsgeschwindigkeit von BLE ist zwar mit 1 MBit/s deutlich höher als z. B. bei NFC (siehe Kapitel 2.4.1), kann aber nicht mit Ethernet, WLAN oder Bluetooth Classic konkurrieren [Wikipedia, 2015]. Allerdings basieren andere, für Multiscreen-Szenarien interessante Protokolle auf BLE, die im folgenden Kapitel vorgestellt werden.

### ***iBeacon und Eddystone***

Aufbauend auf BLE spezifiziert Apple mit *iBeacon* ein proprietäres Protokoll, das es ermöglicht, Distanzen zu anderen Geräten zu messen und diese eindeutig zu identifizieren. Mittels BLE können auch kleine, batteriebetriebene Geräte kontinuierlich ein Bluetooth-Signal aussenden, das sog. *Beacon* (deut. Signal- oder Leuchtfener). Die Nutzdaten dieses Signals werden im iBeacon-Protokoll so spezifiziert, dass sie einen eindeutigen Universally Unique Identifier (UUID) sowie zwei weitere Werte, die Major- und Minor-Nummer, übermitteln können [Apple, 2014]. Es muss kein Pairing der Geräte stattfinden, da alle notwendigen Informationen bereits in den sog. Advertisement-Paketen des BLE-Signals übermittelt werden.

Für ein sendendes Gerät werden zumeist die Begriffe iBeacon und Beacon synonym verwendet, wobei Beacon allgemeiner ist und auch andere Protokolle einschließt.

Store Location		San Francisco	Paris	London
UUID		D9B9EC1F-3925-43D0-80A9-1E39D4CEA95C		
Major		1	2	3
Minor	Clothing	10	10	10
	Housewares	20	20	20
	Automotive	30	30	30

**Abbildung 2.6.:** Nutzung von UUID, Major- und Minor-Nummer zur Realisierung einer Lokalisierung mittels iBeacon. Quelle: [Apple, 2014].



Abbildung 2.6 zeigt exemplarisch, wie mittels dieser Werte Abteilungen in verschiedenen Niederlassungen einer Kaufhauskette identifiziert werden könnten. Die UUID ist weltweit eindeutig für die Organisation, während die Major- und Minor-Nummern für die Niederlassung bzw. die Art der Abteilung verwendet werden. So kann jedes Beacon individuell für den jeweiligen Anwendungsfall konfiguriert werden.

Neben diesen Werten kann ein empfangendes Gerät anhand der Stärke des BLE-Signals den Abstand zu dem Beacon messen, wobei Apple vier verschiedene Abstände unterscheidet [Apple, 2014]:

- *Immediate*: Sehr naher, fast direkter Kontakt zwischen Beacon und Empfänger.
- *Near*: Abstand von ein bis drei Metern.
- *Far*: Abstand von mehr als drei Metern.
- *Unknown*: Der Abstand kann nicht zuverlässig ermittelt werden.

Physikalische Barrieren, ein schwaches BLE-Signal oder hohe Last auf der verwendeten Bluetooth-Frequenz können die Genauigkeit der Abstandsmessung beeinträchtigen.

Google versucht mit *Eddystone* ein offenes Protokoll zu etablieren, das das proprietäre iBeacon ablösen soll und Beacons semantisch wertvollere Daten senden lässt [Google, 2015]:

- *Eddystone-UID*: Eine eindeutige ID analog zur UUID von iBeacon.
- *Eddystone-URL*: Eine kurze URL, die vom Client verwendet werden kann.
- *Eddystone-TLM*: Ein Telemetriepaket zur Überwachung des Zustandes von Beacons.

Neben batteriebetriebenen Sendern ist es möglich, z. B. Smartphones oder Tablets direkt als Beacons zu verwenden. Apple sieht darin zwar nur begrenzte Anwendungsmöglichkeiten [Apple, 2014], für Multiscreen-Szenarien ist allerdings auch diese Variante denkbar.

### **NFC**

Near Field Communication (NFC) ist ein seit 2002 entwickelter Standard zum drahtlosen Datenaustausch zwischen mobilen Endgeräten. Seit 2004 übernimmt das

NFC Forum die Weiterentwicklung des Standards und bietet Informationen zur Entwicklung von NFC-fähiger Software an [NFC Forum, 2015a].

Im Gegensatz zu anderen Drahtlostechnologien wie WLAN oder Bluetooth funktioniert NFC nur in unmittelbarer Nähe der zu verbindenden Geräte (bis zu 20 cm). Dadurch kann der Anwender schon durch räumliche Distanz eine ungewollte Verbindung zu anderen Geräten verhindern. Der Standard ist mit der bereits weit verbreiteten RFID-Technologie kompatibel und kann passive RFID-Tags im 13,56-MHz Band auslesen [Want, 2006, 30]. Die Übertragungsgeschwindigkeit ist auf 424 kBit/s limitiert.

NFC kann in drei verschiedenen Modi betrieben werden (siehe Abbildung 2.7). Im Tag-Reader/Writer-Modus ist es möglich, Tags auszulesen; so können physikalische Objekte mittels extern angebrachter Tags mit elektronisch auslesbaren Informationen versehen werden. Damit können z. B. Fahrpläne um Echtzeitinformationen erweitert werden [NFC Forum, 2015b].

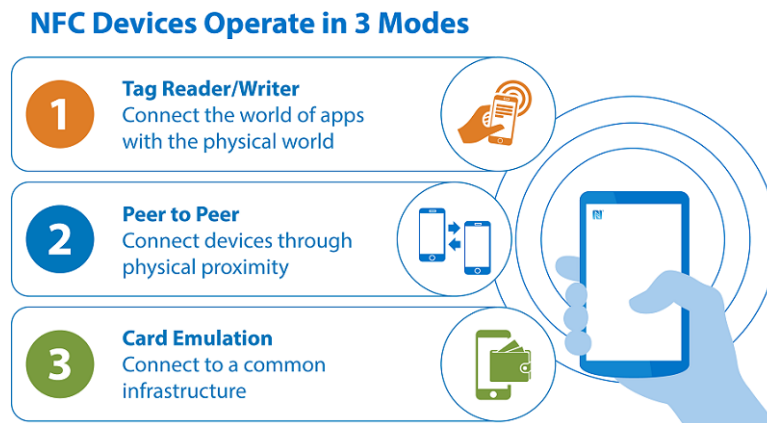
Im Peer-to-Peer-Modus können Geräte direkt miteinander kommunizieren und Daten austauschen, allerdings ist die Bandbreite eine entscheidende Limitierung für die Art der auszutauschenden Daten. Im Card Emulation Modus kann sich ein NFC-fähiges Gerät als Smart Card ausgeben und in bestehende Smart Card Infrastrukturen integriert werden [NFC Forum, 2015b].

Vor allem die ersten beiden Modi sind für Multiscreen-Szenarien interessant, um Geräte zu verbinden, kleinere Datenmengen auszutauschen oder physikalische Objekte in Interaktionen mit einzubeziehen. Die meisten mobilen Geräte sind heute mit NFC ausgestattet, was den Einsatz besonders für diese Gerätekategorie attraktiv macht.

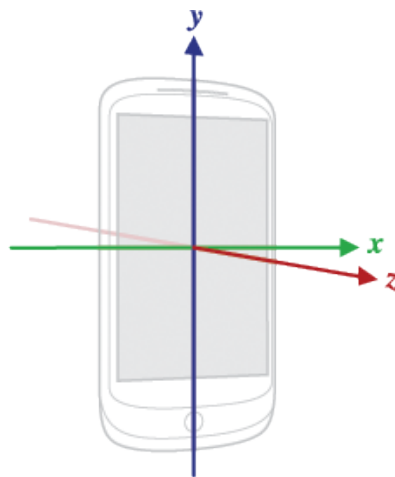
### 2.4.2. Sensoren

#### ***Beschleunigungssensor***

Beschleunigungssensoren (engl. accelerometer) messen die Beschleunigung, der sie zum Zeitpunkt der Messung ausgesetzt sind. In vielen mobilen Geräten sind Beschleunigungssensoren verbaut, die entlang der drei Achsen des Gerätes Beschleunigungsvektoren messen können. Dabei wird das Standardkoordinatensystem aus Abbildung 2.8 verwendet. Ein einfacher Anwendungsfall wäre z. B. eine Applikation, die das Schütteln des Gerätes durch eine alternierende Beschleunigung erkennt.



**Abbildung 2.7.:** Die drei Betriebsmodi von NFC. Quelle: [NFC Forum, 2015b].



**Abbildung 2.8.:** Das Standardkoordinatensystem für Sensoren mobiler Geräte. Quelle: [Android, 2015c].

Die Beschleunigungswerte werden mit einer bestimmten *Frequenz* gemessen, die zwischen verschiedenen Sensoren und Implementierungen variieren kann. Ebenso kann sich die *Auflösung* unterscheiden, also die Genauigkeit, mit der die Werte ermittelt werden. Einen Einfluss auf die Messwerte hat die Erdbeschleunigung von  $g = 9.81 \text{ m/s}^2$ , die je nach Orientierung des Gerätes auf eine oder mehrere Achsen wirkt. Ein anschauliches Beispiel dazu findet sich in der Android-Dokumentation. Angenommen wird ein Gerät, das flach auf einem Tisch liegt [Android, 2015c]:

- Beim Beschleunigen des Gerätes von links (das Gerät bewegt sich nach rechts): positiver Beschleunigungswert auf der X-Achse.
- Beim Beschleunigen des Gerätes von unten (das Gerät bewegt sich nach oben): positiver Beschleunigungswert auf der Y-Achse.

## 2. Technische und konzeptionelle Grundlagen

---

- Beim Aufnehmen des Gerätes mit einer Beschleunigung von  $x \text{ m/s}^2$  wird auf der Z-Achse ein Wert von  $a - g = a - (-9.81 \text{ m/s}^2) = a + 9.81 \text{ m/s}^2$  gemessen, also der Beschleunigung des Gerätes plus die Erdanziehung.
- Bei ruhiger Lage des Gerätes auf dem Tisch wird eine Beschleunigung von  $0 \text{ m/s}^2 - (-9.81 \text{ m/s}^2) = 9.81 \text{ m/s}^2$  auf der Z-Achse gemessen.

Mit Hilfe eines *linearen Beschleunigungssensors* können die Beschleunigungen entlang der Achsen auch exklusive der Erdanziehung gemessen werden, ohne sie durch Filter nachträglich entfernen zu müssen [Android, 2015a].

Erweiterte Anwendungen des Beschleunigungssensors finden sich bei Roy und Kollegen, die einen Beschleunigungssensor zum Dekodieren von modulierten Vibrationsignalen verwenden [Roy u. a., 2015], oder in [Hinckley u. a., 2004], wo das Zusammenschlagen („Bump“) zweier Tablets mittels Beschleunigungssensoren erkannt wird.

### **Gyroskop**

In vielen mobilen Geräten ist neben dem (linearen) Beschleunigungssensor auch noch ein Drehratensensor, meist *Gyroskop* genannt, verbaut. Damit kann die Winkelgeschwindigkeit des Gerätes um die drei Achsen (siehe Abbildung 2.8) gemessen werden. Während der Beschleunigungssensor die Beschleunigung *entlang* der Achse misst, misst das Gyroskop die Geschwindigkeit *um* die Achse.

Die Rohwerte beschreiben die Rotationsrate in rad/s, wobei meist eine Integration der Werte vorgenommen wird, um Winkeländerungen über feste Zeitfenster hinweg zu bestimmen. Zudem gibt es Störfaktoren, die durch die Kombination des Gyroskops mit anderen Sensoren korrigiert werden können [Android, 2015a].

Ein anschaulicher Anwendungsfall für das Gyroskop wäre z. B. eine Spin-the-bottle-App, in der die Drehung des Smartphones um die Z-Achse eine Flaschendrehung simuliert. Schon Rekimoto machte 1996 Gebrauch von Gyroskopen in tragbaren Geräten, um durch leichte Neigung des Gerätes Menüeinträge durchzuschalten [Rekimoto, 1996].

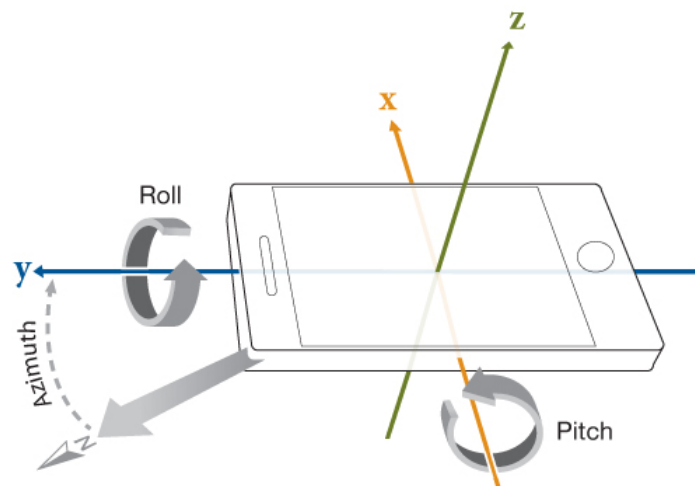
Das Gyroskop misst nur die Änderungsrate um die Achsen des Gerätes ohne weiteren Kontext. Im nächsten Kapitel wird gezeigt, wie auch die Lage des Gerätes im Raum bestimmt werden kann.

### **Magnetometer und Orientierung**

Um zu bestimmen, welche Lage ein Gerät relativ zu einem Referenzkoordinatensystem hat, kann ein Orientierungssensor benutzt werden. Dabei handelt es sich um einen Softwaresensor, der die Werte des in Kapitel 2.4.2 beschriebenen Beschleunigungssensors mit den Werten eines *Magnetometers*, der Änderungen in der Stärke des magnetischen Feldes der Erde misst, kombiniert.

Durch Kombination und Transformation dieser Werte können drei Werte ermittelt werden [Android, 2015b]:

1. *Azimuth*: Winkel zwischen Y-Achse des Gerätes und dem magnetischen Nordpol. Zeigt die Y-Achse zum magnetischen Nordpol, ist der Wert  $0^\circ$ , für Süden  $180^\circ$ , für Osten und Westen  $90^\circ$  bzw.  $270^\circ$ .
2. *Pitch*: Rotationswinkel um die X-Achse des Gerätes. Wird die positive Z-Achse in Richtung der positiven Y-Achse bewegt (siehe Abbildung 2.9), ist der Wert positiv, sonst negativ.
3. *Roll*: Rotation um die Y-Achse des Gerätes. Funktioniert analog zu Pitch, allerdings bezogen auf die X- und Z-Achse des Gerätes.



**Abbildung 2.9.:** Azimuth, Pitch und Roll bei mobilen Geräten. Quelle: [MathWorks, 2014].

Im Gegensatz zu Beschleunigungssensor und Gyroskop beschreiben diese Werte eine Lage im Raum, keine Änderungsrate. Dabei ist der Azimuth-Wert interessant, um z. B. Kompass-Apps zu realisieren. Eine Kombination aller drei Werte kann genutzt werden, um die Lage verschiedener Geräte zueinander zu bestimmen.

### **Touchscreens**

Ein Nutzer mit Desktop-PC wird seine Eingaben üblicherweise mit Maus und Tastatur machen. Laptops haben die Maus durch ein Touchpad bzw. einen Trackpoint ersetzt, womit der Nutzer Bewegungen des Mauszeigers und teilweise auch Tastendrucke oder einfache Gesten durchführen kann.

Mit dem iPhone hat Apple als erste Firma ein mobiles Gerät mit Touchscreen entwickelt, das die Bedienbarkeit deutlich interaktiver und simpler gestaltete als Konkurrenzprodukte, deren Design noch immer im Desktop-Stil gehalten wurde. Dadurch wurden auch Gesten wie z. B. *pinch* zum Vergrößern von Fotos ermöglicht [Banga und Weinhold, 2014, 42]. Heute besitzen in der Regel alle Smartphones und Tablets einen Touchscreen, aber auch größere Geräte werden inzwischen mit Touchscreens ausgestattet.

Es gibt verschiedene Technologien, um Touchscreens zu realisieren. Walker unterscheidet zwischen sechs Basistechnologien von Touchscreens [Walker, 2015, 35]:

1. kapazitiv
2. resistiv
3. akustisch
4. optisch
5. eingebettet
6. andere

Jede dieser Technologien hat verschiedene Vor- und Nachteile. Eines der Schlüssel-features für Touchscreens ist die *Multitouch*-Fähigkeit, also die Möglichkeit, mehrere simultane Berührungen auf einem Touchscreen gleichzeitig zu erkennen [Walker, 2015, 39].

Kapazitive Displays, die in den meisten mobilen Geräten verbaut sind, besitzen theoretisch keine Limitierung für die Anzahl der Multitouch-Punkte, sind aber in der Skalierung der Größe begrenzt. Große Displays wie der Microsoft PixelSense sind mit Infrarot-Touchscreens ausgestattet, die zwar 32 und mehr Multitouch-Punkte erkennen und in der Größe gut skalieren, dafür aber anfällig für Störungen bei heller Beleuchtung (z. B. unter freiem Himmel) sind.

Resistive Displays erkennen im Gegensatz zu kapazitiven auch die Berührung mit einem Stylus oder anderen spitzen Objekten, dafür fehlt hier die Multitouch-Unter-

stützung. Zudem ist mehr Kraftaufwand beim Berühren des Touchscreens erforderlich [Walker, 2015, 47,56,76].

Bei der Gestaltung von Multiscreen-Interaktionen, die auf Touchscreens basieren, müssen diese Einschränkungen berücksichtigt werden. Für die Eingabe längerer Texte auf Touchscreens gibt es zwar verschiedene Lösungen, allerdings konnten mobile Geräte Desktop-PCs mit Keyboard und Maus im Sinne benutzerfreundlicher Interaktion für diesen Fall bisher nicht ablösen. [Banga und Weinhold, 2014, 46].

### **Audio und Video**

Mobile Geräte sind in der Regel mit Mikrofonen und Kameras ausgestattet, für PCs und Smart TVs können diese günstig zugekauft werden. Übliche Nutzungsszenarien für Mikrofone und Kameras sind die Aufnahme von Bildern, Videos und Tondateien oder (Video)-Telefonie.

In Kapitel 2.4.3 wird gezeigt, wie Shirazi die Kamera als Sensor einsetzt, um modulierte Lichtsignale zu empfangen und eine entsprechende Systemreaktion auszulösen. Aumi und Kollegen demonstrieren mit ihrem DopLink-System die Möglichkeit, mittels Mikrofon einen Dopplereffekt zu erkennen. Dadurch kann die Bewegung eines mobilen Gerätes erkannt werden, das ein Ultraschallsignal aussendet. Über diesen Mechanismus wird die Verbindung zwischen den Geräten aufgebaut [Aumi u. a., 2013].

### **2.4.3. Ausgabe**

Im Gegensatz zu den vielfältigen Sensoren zur Messung von Nutzeraktionen in Kapitel 2.4.2 ist das Feld möglicher Ausgaben für Systemreaktionen überschaubar.

Alle Geräteklassen ermöglichen schon per Definition in Kapitel 2.1 eine Bildschirmausgabe. Zwar können Größe und Form des Bildschirms stark variieren, visuelles Feedback ist jedoch generell möglich und in der Regel mittels der jeweiligen GUI-Frameworks einfach zu realisieren.

Fast ebenso universell vorhanden ist die Möglichkeit für akustisches Feedback. Als Kommunikations- bzw. Unterhaltungsgeräte besitzen Geräte der Klassen Smartphone und Tablet in aller Regel eine Audioausgabe. Auch Laptops und kleinere Smart TVs sind mit Audioequipment ausgestattet. Lediglich Desktop PCs oder große Leinwände können unter Umständen kein akustisches Feedback liefern.

Mobile Geräte sind zudem häufig mit einem Vibrationsmotor ausgestattet, der in Geräten der anderen Klassen meist nicht vorhanden ist. Vibration kann als Systemreaktion eingesetzt werden, um z. B. Tastendrucke auf einem Touchscreen mit einer gefühlten Haptik zu versehen. Oliveira und Kollegen haben gezeigt, dass sich damit sogar eine Braille-Tastatur mittels Touchscreens realisieren lässt [Oliveira u. a., 2011]. Zudem kann Vibration in Kombination mit anderen Ausgaben eingesetzt werden, z. B. als Ergänzung zu akustischem Feedback, um in lauten Umgebungen trotzdem die Aufmerksamkeit des Nutzers zu erlangen. Für Geräte, die sich nicht in unmittelbarer Nähe des Nutzers befinden, ist diese Art von Feedback nicht nutzbar.

Tablets und Smartphones sind teilweise mit einer ansteuerbaren Lampe ausgestattet. Neben typischen Anwendungsfällen wie Taschenlampen- oder Foto-Apps haben Shirazi und Kollegen gezeigt, wie ein Anwender mittels seiner Smartphone-Lampe einen Pointer auf einem großen Display steuern kann, dessen Kamera das Lichtsignal aufnimmt und verarbeitet [Shirazi u. a., 2009]. Nichtsdestotrotz sind die Einsatzmöglichkeiten in Multiscreen-Interaktionen eher eingeschränkt.



## Kapitel 3

# Modell für Multiscreen-Interaktionen

### 3.1. Herausforderungen

Die in Kapitel 2.3 vorgestellte Patternvorlage dient als Basis, das Wissen zu den einzelnen Multiscreen-Patterns einheitlich und strukturiert erfassen zu können. Während Aspekte wie Problem, Lösung oder Anwendungskontext mit natürlicher Sprache bzw. mittels Checkboxes und Listen erfasst werden können, ist es deutlich schwieriger, eine konsistente, strukturierte Form für die Erfassung der technischen Realisierung von Patterns, also der Gestaltung des „Wie“, zu finden.

Für verschiedene Patternsammlungen wurde bei der Dokumentation der technischen Umsetzung dabei sehr unterschiedlich vorgegangen. Gamma und Kollegen verwenden eine Mischung aus OMT<sup>1</sup>- und Interaktionsdiagrammen sowie Code-Beispielen in C++ und SmallTalk, um die strukturelle und semantische Umsetzung der Patterns zu dokumentieren [Gamma u. a., 1995]. Durch den Einsatz gängiger Programmier- und Modellierungssprachen sind die Patterns für Entwickler einfach nachzuvollziehen und umzusetzen.

Die Patternsammlungen von Tidwell [2010], Yahoo [2015] und van Welie [2008], die sich mit der Lösung wiederkehrender Probleme bei der Gestaltung von Applikations- und Webinterfaces befassen, nutzen natürlichsprachliche Beschreibungen, meist ergänzt um Beispiele in Form von Screenshots. Auf Formalismen zur abstrakten Beschreibung von GUI- und Interaktionsdesign wird hier gänzlich verzichtet, da die Patterns keine strikten Regeln und keine Schritt-für-Schritt Anleitungen für das Interface-Design sein sollen [Tidwell, 2010, xviii].

---

<sup>1</sup>Die Object Modeling Technique (OMT) ist eine Anfang der Neunzigerjahre entwickelte Objekt-Modellierungssprache.

Bei der Beschreibung der technischen Aspekte von Multiscreen-Patterns ergeben sich besondere Herausforderungen. Neben der GUI-Gestaltung müssen auch Interaktionsmodalitäten, Kommunikationsprotokolle oder technische Voraussetzungen, unter denen der Einsatz bestimmter Patterns überhaupt erst möglich ist, berücksichtigt werden. Zudem steht im Zentrum jedes Multiscreen-Patterns eine oft komplexe Geste, deren Erkennungsalgorithmus beschrieben werden muss. Daraus ergibt sich, dass Multiscreen-Patterns als Kommunikationsmittel zwischen Designern, Entwicklern, Softwarearchitekten und anderen Projektbeteiligten verstanden und genutzt werden müssen. Dabei sollten einige zentrale Fragen durch die technische Beschreibung des Patterns beantwortet werden, wie bspw.:

- Kann das Pattern in einer gegebenen Infrastruktur eingesetzt werden?
- Gibt es technische Einschränkungen bei der Umsetzung des Patterns?
- In welcher Form interagiert der Nutzer mit dem Gerät?
- Was für Eingaben kann der Nutzer tätigen, welche Systemreaktionen gibt es?
- Welche Systemkomponenten können an welcher Stelle wiederverwendet / kombiniert werden?
- Wie sieht das Kommunikationsprotokoll aus, das die beteiligten Geräte implementieren müssen?
- Welche Daten werden übermittelt?

Um ein einheitliches Modell zu schaffen, das die Beantwortung dieser Fragen für jedes Pattern ermöglicht, wird sich die Analyse in diesem Kapitel mit Publikationen zum Thema Multiscreen-Interaktion auseinandersetzen, die für verschiedene Multiscreen-Szenarien bereits eine theoretische oder praktische technische Realisierung beschreiben. Für diese heterogenen Konzepte wird anschließend eine gemeinsame Abstraktion erarbeitet. Als Leitlinie für die Analyse und Modellierung werden im folgenden Kapitel zunächst konkrete Anforderungen an das Modell formuliert.

## 3.2. Anforderungen an das Modell

Aus dem Projektziel von SysPlace und den zentralen Forschungsfragen in Kapitel 1.1 sowie den bereits erarbeiteten Patterns in Kapitel 2.3 lassen sich Anforderungen

an das Modell aus Sicht verschiedener Nutzergruppen ableiten. Wie in Kapitel 2.3 beschrieben, zerfällt ein Multiscreen-Pattern in die Durchführung einer Geste und eine darauf folgende Systemreaktion. Die konzeptionelle Erfassung von Gesten entsprechend der ersten Forschungsfrage in Kapitel 1.1 ist durch das Patterntemplate bereits vorgenommen worden (vgl. Anhang A).

Soll ein Pattern in einer Multiscreen-Anwendung realisiert werden, wird sich für den Entwickler die Frage stellen, wie die entsprechende Geste technisch erkannt werden kann. Das Patterntemplate in Anhang A beschreibt unter „Wie“ zwar allgemein die Durchführung der Geste und unter „Technisches“ mögliche Technologien zur Erkennung von Gesten, trifft allerdings keine Aussage darüber, wie diese Technologien konkret eingesetzt und ggf. kombiniert werden müssen. Zudem werden technische Alternativen für die Erkennung nicht ausreichend berücksichtigt.

Folgende Aspekte sollen durch das Modell daher hinsichtlich der zweiten<sup>2</sup> und dritten<sup>3</sup> Forschungsfrage beschrieben werden:

- Die zu verarbeitenden Eingaben des Nutzers in Form von Sensordaten oder anderen Systemevents;
- Hard- und Softwarevoraussetzungen, unter denen diese Daten verfügbar sind;
- Abstrakte Beschreibung des Erkennungsalgorithmus einer Geste auf Basis von Sensordaten;
- Alternative Erkennungsmöglichkeiten und deren Implikationen;
- Möglichkeiten zur Einschränkung, unter welchen Umständen eine Geste erkannt wird;
- Kommunikationsprotokolle zwischen Geräten für die Erkennung geräteübergreifender Gesten;
- Wiederverwendbarkeit von Teilkomponenten.

Diese Aspekte sollten unabhängig von konkreten Technologien modelliert werden, sodass eine Umsetzung auf verschiedenen Plattformen möglich ist. Zudem soll eine Repräsentation gewählt werden, deren Syntax und Regeln einem Entwickler intuitiv verständlich sind.

---

<sup>2</sup>Welche methodisch-/technischen Voraussetzungen sind notwendig, um diese Interaktionen zu unterstützen?

<sup>3</sup>Welche Entwicklungsverfahren, -werkzeuge und -modelle und Voraussetzungen muss die zugrunde liegende Softwarearchitektur des Gesamtsystems mitbringen, um die Szenarien zu unterstützen?

### 3. Modell für Multiscreen-Interaktionen

---

Nachdem eine Geste durch das System erkannt wurde, folgt eine Systemreaktion entsprechend der fünf Kategorien<sup>4</sup> aus Kapitel 2.3. Dabei handelt es sich meist nicht um einfache Reaktionen wie z. B. Textausgaben auf dem Bildschirm, sondern um komplexere Abläufe, die eventuell weitere Ein- und Ausgaben oder Datenverarbeitung beinhalten. Eine allgemeine Beschreibung der Systemreaktion sowie möglicher Erfolgs- und Fehlerfälle ist im Patterntemplate unter „Wie“ zu finden, allerdings wird hier nur ein konzeptioneller Überblick gegeben, der Detailgrad reicht für die Beantwortung der Forschungsfragen zwei<sup>5</sup> und drei<sup>6</sup> nicht aus.

Aus Sicht eines Interaktionsdesigners sollten daher folgende Aspekte modelliert werden:

- Vollständige Ablaufbeschreibung der Systemreaktion,
- eventuelle Vor- und Nachbedingungen der Systemreaktion,
- mögliche Interaktionen des Nutzers,
- mögliche Rückmeldungen des Systems an den Nutzer und
- Implikationen der gewählten Geste auf die Abläufe der Multiscreen-Anwendung.

Jede Kategorie hat eigene Implikationen, die von dem Modell erfasst werden sollten, wie z. B. uni- und bidirektionale Datenübertragung oder der Verbindungsaufbau zwischen Geräten. Interaktionsdesigner gestalten diese Abläufe zwar konzeptionell, die technische Realisierung aber übernehmen Entwickler. Aus Entwicklersicht kommen noch technische Aspekte hinzu, die im Modell berücksichtigt werden müssen:

- Benötigte Objekte und deren Datenfluss im System,
- Implikationen verarbeiteter Daten auf den weiteren Programmfluss,
- Zustand vor und nach der Systemreaktion und
- Beschreibung möglicher Alternativen oder Fehlerfälle.

---

<sup>4</sup>*Connect* (Geräte Verbinden), *Give* (Daten übermitteln), *Take* (Daten empfangen), *Exchange* (Daten austauschen), *Extend* (Geräte koppeln).

<sup>5</sup>Welche methodisch-/technischen Voraussetzungen sind notwendig, um diese Interaktionen zu unterstützen?

<sup>6</sup>Welche Entwicklungsverfahren, -werkzeuge und -modelle und Voraussetzungen muss die zugrunde liegende Softwarearchitektur des Gesamtsystems mitbringen, um die Szenarien zu unterstützen?

Da das Modell an dieser Stelle ein wichtiges Kommunikationsmittel zwischen Entwicklern und Interaktionsdesignern darstellt, soll eine für beide verständliche Repräsentation eingesetzt werden.

Zusätzlich zu den Anforderungen, die sich aus dem Patterntemplate ergeben, sind noch weitere, generelle Aspekte für die Modellierung relevant. Die Erfassung und Überarbeitung von Patterns ist ein kontinuierlicher Prozess. Es ist möglich, dass weitere Patterns in die Sammlung aufgenommen oder alternative Lösungsansätze erfasst werden. Das Modell sollte daher nicht nur den aktuellen Umfang der Sammlung abbilden können, sondern offen für Erweiterungen innerhalb des Kontexts der Multiscreen-Interaktion sein.

Per Definition bieten Patterns keine fertige Lösung für ein konkretes Problem, sondern abstrakte Lösungsansätze für wiederkehrende Problemtypen (vgl. Kapitel 2.3). In der Praxis bedeutet das, dass Anwender von Patterns eine Instanz des Modells erstellen müssen, die eine Realisierung der vorhandenen Anwendungsfälle ermöglicht. Das Projektziel in Kapitel 1.1 definiert als Anwender der Patterns kleine und mittelständische Unternehmen. Damit diese mit dem Modell arbeiten können, sollten die eingesetzten Modellierungssprachen und -werkzeuge frei verfügbar sein und kein umfangreiches Vorwissen benötigen.

Je nachdem in welchem Multiscreen-Szenario Patterns eingesetzt werden, sind eventuell Geräte bestimmter Klassen vorhanden. Das Modell sollte diesen Punkt berücksichtigen und die Beziehung zwischen einem konkret modellierten Lösungsansatz und den dafür benötigten Mindestanforderungen an Hard- und Software abbilden.

### **3.3. Erarbeitung des Modells**

Die folgenden Unterkapitel beschreiben schrittweise das erarbeitete Modell. Dabei wird jeweils nach demselben Schema vorgegangen:

1. Kurzer Überblick über den Fokus des Teilmodells und die Einordnung in das Gesamtmodell,
2. Präsentation und Erklärung des erarbeiteten Teilmodells und
3. ausführliche Begründung für die Modellierungsentscheidungen.

Zur Modellierung werden Diagramme der Unified Modeling Language (UML) in der aktuellen Version 2.5 verwendet. Die UML dient zur Modellierung, Dokumentation, Spezifizierung und Visualisierung komplexer Systeme, unabhängig von deren Fachdomäne. Sie liefert Notationselemente für die statischen und dynamischen Modelle von Analyse, Design und Architektur und unterstützt insbesondere objekt-orientierte Vorgehensweisen [Rupp u. a., 2012, 4].

Laut Rupp und Kollegen ist die UML zudem [Rupp u. a., 2012, 4]

- nicht vollständig,
- keine Programmiersprache,
- keine rein formale Sprache,
- kein vollständiger Ersatz für Textbeschreibung und
- weder Methode noch Vorgehensmodell.

All diese Eigenschaften passen zu den Herausforderungen und Anforderungen an das Modell (vgl. Kapitel 3.1 und 3.2). UML ermöglicht es, die technische Komplexität von Multiscreen-Patterns zu visualisieren und dadurch die Textbeschreibungen um detaillierte Diagramme zu ergänzen. Zudem sind mit StarUML, UMLet, Eclipse etc. viele frei verfügbare Werkzeuge für die UML-Modellierung vorhanden. Die weite Verbreitung von UML trägt dazu bei, dass die gängigen Diagrammtypen von den meisten Entwicklern gelesen werden können.

Verschiedene Aspekte des Modells werden mit dem jeweils geeigneten Diagrammtyp dargestellt. Der UML-Standard legt nicht exakt fest, welche Elemente in welchem Diagrammtypen vorkommen dürfen [Rupp u. a., 2012, 15]. Bevor in dieser Arbeit ein Diagrammtyp erstmals verwendet wird, wird kurz beschrieben, in welcher Form er verwendet wird und wo eventuelle Abweichungen zur gängigen Verwendung von UML vorgenommen wurden.

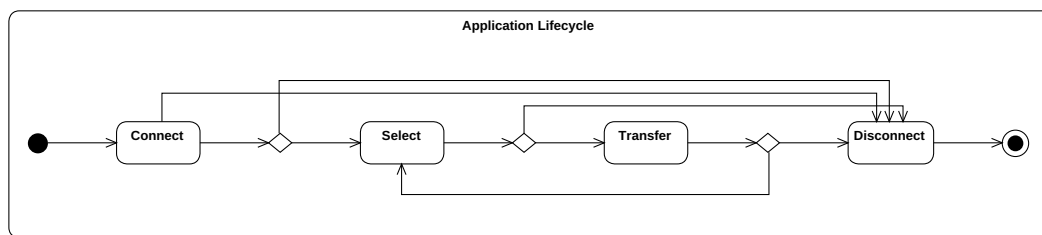
Das Ergebnis ist ein *semi-formales* Modell, das *visuell* repräsentiert wird und *statische* sowie *dynamische Aspekte* der Multiscreen-Interaktion abbildet.

#### 3.3.1. Rahmenmodell

Die SysPlace-Patterns beschreiben zwar jeweils nur eine Geste mit einer darauf folgenden komplexen Systemreaktion (siehe Kapitel 2.3), müssen aber bei der Realisierung stets in einen Applikationskontext eingebettet werden. Bei näherer Betrachtung

tung der fünf Kategorien fällt zudem auf, dass hier zwar keine *Pattern Language*<sup>7</sup> vorliegt, da alle Patterns das gleiche Abstraktionsniveau haben und es keine hierarchische Struktur zwischen den Patterns gibt, die Patterns der Kategorie „Connect“ allerdings nur zur Verbindung von Geräten genutzt werden, während die Patterns der anderen vier Kategorien in irgendeiner Weise Daten übertragen und somit eine bereits bestehende Verbindung voraussetzen. Dadurch entsteht eine gewisse Abhängigkeit zwischen den Pattern dieser Kategorien.

Das Rahmenmodell aus Abbildung 3.1 beschreibt einen Applikationslebenszyklus aus Sicht von Multiscreen-Interaktionen, der die Einsatzgebiete der Patterns berücksichtigt und in Beziehung zueinander setzt. Am Anfang steht immer eine Aktivität zur Verbindung von Geräten (*Connect*), auf die entweder eine sofortige Trennung (*disconnect*) oder eine beliebig oft wiederholte Abfolge von Auswahl- und Transfer-Aktivitäten (*Select* bzw. *Transfer*) folgt, bis irgendwann die Verbindung getrennt und der Anwendungslebenszyklus beendet wird. Verschiedene Systeme und Kon-

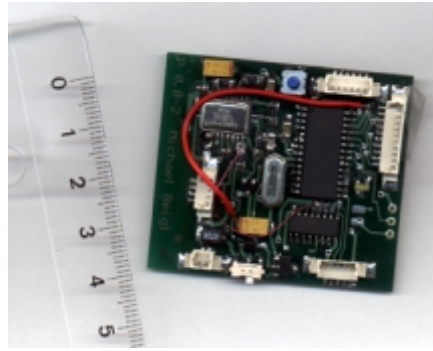


**Abbildung 3.1.:** Rahmenmodell für Multiscreen-Interaktion.

zepte, die Multiscreen-Szenarien realisieren, implementieren diesen Ablauf komplett oder in Teilen und bilden die Grundlage für die Erstellung des Modells. Die wichtigsten Konzepte werden im Folgenden näher betrachtet.

Holmquist und Kollegen stellen das System *Smart-Its Friends* vor, mit dem Geräte um eine externe Komponente erweitert werden können (siehe Abbildung 3.2), die einen impliziten (durch Annäherung) oder expliziten (durch eine Schüttel-Geste) Verbindungsaufbau zwischen diesen Geräten ermöglicht [Holmquist u. a., 2001]. Nachdem eine direkte Verbindung zwischen den Geräten aufgebaut wurde, bleibt es der Anwendung überlassen, diesen Kommunikationskanal zur Übertragung anwendungsspezifischer Nutzdaten zu verwenden. Der Verbindungsabbau findet entweder statt, wenn verbundene Geräte sich nicht mehr in Sendereichweite befinden oder wenn die Applikation die Verbindung aktiv schließt.

<sup>7</sup>Borchers beschreibt eine Pattern Language als einen gerichteten Graphen, in dem Patterns höherer Ebenen auf Patterns niedrigerer Ebenen verweisen, wodurch sich eine Hierarchie von gemeinsam nutzbaren Patterns ergibt [Borchers, 2000].



**Abbildung 3.2.:** Smart-Its Friends Sender / Empfänger. Quelle: [Holmquist u. a., 2001].

Aus Sicht von Smart-Its Friends erfolgt also immer zuerst der Verbindungsschritt, also die Aktivität *Connect*. Danach kann entweder ein *Disconnect* stattfinden (wenn die Geräte auseinander bewegt werden) oder es ist möglich, dass die Anwendung solange Daten auswählt und transferiert, bis ein Zustand erreicht wird, in dem der *Disconnect* durch die Anwendung erfolgt. Diese möglichen Abläufe finden sich im Modell in Abbildung 3.1 wieder.

Dachselt und Buchholz entwerfen ein Multiscreen-Szenario, das einen ähnlichen Ablauf realisiert, obwohl andere Technologien und Gesten eingesetzt werden [Dachselt und Buchholz, 2009]. Die Interaktion findet hier zwischen einem Smartphone und einem Smart TV statt (siehe Abbildung 3.3). Betritt ein Anwender den Raum, wird automatisch eine Wi-Fi- oder Bluetooth-Verbindung zwischen den beiden Geräten hergestellt. Eine Wurfgeste des Smartphones in Richtung des Smart TVs überträgt die Bilder des Smartphones auf das TV-Gerät. Anschließend können die einzelnen Bilder auf dem Smart TV durch Kippen (*Tilt-Geste*<sup>8</sup>) des Smartphones durchgeschaltet werden. Eine ruckartige Geste zum Körper (*Fetch-Back-Geste*) ermöglicht den Transfer von Bildern in die andere Richtung (vom Smart TV auf das Smartphone), das Verlassen des Raumes schließt die Verbindung.

Auch dieser Ablauf lässt sich durch das Modell beschreiben:

- Aufbauen einer Verbindung durch Betreten des Raumes (*Connect*),
- Auswahl zu übertragender Dateien (*Select*),
- Wurf-Geste zum Anstoßen der Übertragung (*Transfer*),
- Wiederholte Tilt-Gesten zum Durchschalten der Fotos (*Select / Transfer*),

---

<sup>8</sup>Eine ausführliche Beschreibung der Tilt-Geste zum Steuern von Menüs findet sich in [Rekimoto, 1996].

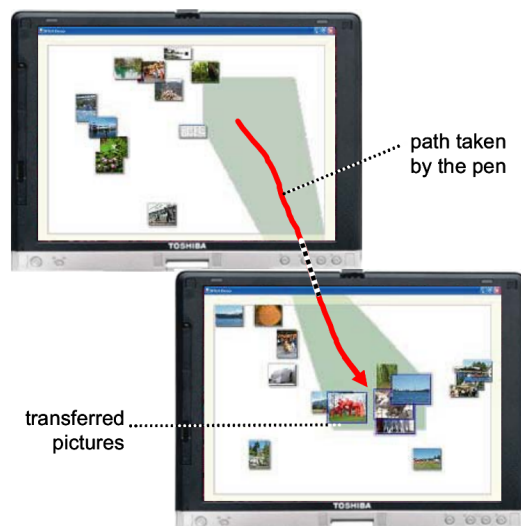




**Abbildung 3.3.:** Wurf-Geste zur Interaktion zwischen Smartphone und Smart TV. Quelle: [Dachselt und Buchholz, 2009].

- Fetch-Back-Geste zum Empfangen von Daten des TV-Gerätes (*Select / Transfer*) und
- Abbauen der Verbindung durch Verlassen des Raumes (*Disconnect*).

In diesem Ablauf gibt es eine Wiederholung der Schritte *Select* und *Transfer*, was im Modell in Abbildung 3.1 berücksichtigt wird. Nach jedem Transfer wartet das Gerät auf weitere Eingaben, erst das Verlassen des Raumes beendet den Ablauf. Zudem gibt es hier noch eine weitere Ergänzung. Die Tilt-Geste überträgt statt Daten des Nutzers nur einen Steuerbefehl, der das Smart TV zum Weiterschalten der Fotos veranlasst. Die Selektion erfolgt hier implizit, weil durch die Geste nur der Zustand auf dem Smart TV verändert wird, aber keine neuen Fotos übertragen werden. Auf diese Details wird in den Kapiteln 3.3.5 und 3.3.6 näher eingegangen.

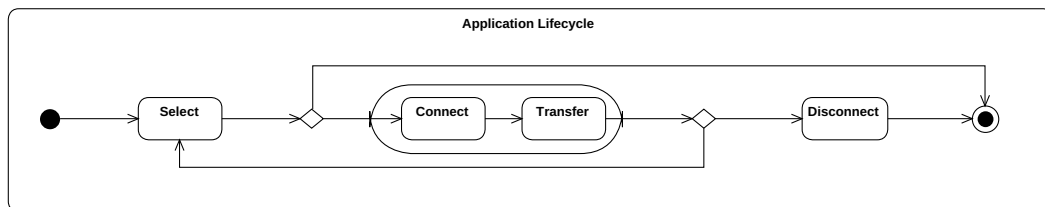


**Abbildung 3.4.:** Stich-Geste zur Übertragung von Bildern. Quelle: [Hinckley u. a., 2004].

Eine weitere wichtige Ergänzung findet sich im System *StitchMaster*, das Hinckley und Kollegen entworfen haben [Hinckley u. a., 2004]. Auch hier wird eine Interak-

tion zwischen zwei Geräten, diesmal mittels einer sog. *Stitch*-Geste, realisiert. Es werden zwei Geräte nebeneinander positioniert und mit einer Stift-Bewegung von einem auf den anderen Bildschirm verbunden (siehe Abbildung 3.4). Die Selektion von Daten findet vor der Verbindung und Übertragung statt. Durch die anschließende *Stitch*-Geste wird sowohl die Verbindung zwischen den Geräten aufgebaut als auch die Übertragung von Daten eingeleitet. Die auszuführende Aktion auf dem Zielgerät wird entweder aus der Art der ausgewählten Daten abgeleitet oder durch den Nutzer auf dem Zielgerät explizit ausgewählt. Der Nutzer hat außerdem die Möglichkeit, eine bestehende Verbindung über die Auswahl eines Menüpunkts zu trennen.

Für das Modell kann aus dem System *StitchMaster* abgeleitet werden, dass bei Verwendung derselben Geste für *Connect* und *Transfer* auch ein alternativer Ablauf eingesetzt werden kann, der in Abbildung 3.5 zu finden ist. Die Aktivitäten *Connect* und *Transfer* sind direkt verbunden und finden gemeinsam statt, die Aktivität *Select* ist beiden vorangestellt. Ein wiederholtes Ausführen der Geste würde die Kombination aus Verbindung und Übertragung erneut anstoßen. Ein ähnlicher Ablauf findet sich auch in [Hinckley, 2003] für *Bump*-Gesten. Die Bedingung, dass dieselbe Geste zum Verbinden und Übertragen zum Einsatz kommt, ist auch hier erfüllt.



**Abbildung 3.5.:** Rahmenmodell mit zusammengefassten *Connect*- und *Transfer*-Aktivitäten.

Jede der vier Aktivitäten des Rahmenmodells kann durch Gesten unterstützt werden, wobei die SysPlace-Patterns vor allem die Gestaltung der Aktivitäten *Connect*, *Transfer* und *Disconnect* adressieren. Im folgenden Kapitel wird zunächst die erarbeitete Modellierung der Gestenerkennung beschrieben, bevor die einzelnen Aktivitäten und deren Modellierung im Detail besprochen werden, da diese auf der Gestenerkennung aufbauen.

#### 3.3.2. Erkennung einfacher Gesten

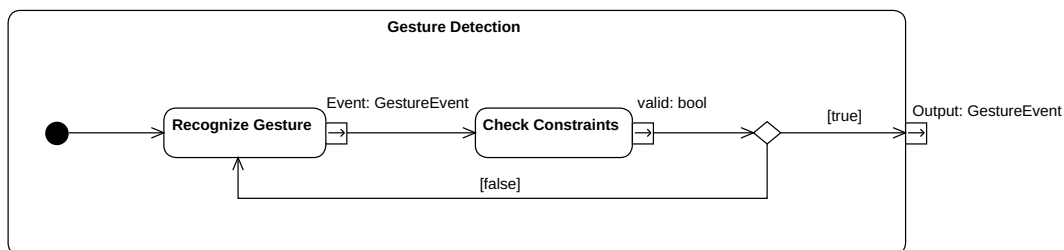
Wie in Kapitel 2.3 beschrieben, besteht ein Multiscreen-Pattern aus einer Geste des Nutzers sowie der ausgelösten Systemreaktion. Das SysPlace-Patterntemplate (sie-

he Anhang A) sieht zwar eine textuelle und grafische Beschreibung von Gesten vor, gibt allerdings wenig Hinweise über Implementierungsdetails. Um eine Gestenerkennung implementieren zu können, müssen dem Entwickler verschiedene Aspekte bekannt sein:

- Benötigte Eingabe- und Sensordaten
- Interpretation / Verarbeitung der Daten
- Mögliche Einschränkungen für die Erkennung
- Kommunikationsprotokoll für geräteübergreifende Gesten

Soll z. B. eine Swipe-Geste erkannt werden, muss ein Touchscreen vorhanden sein, dessen Berührungspunkte ausgelesen werden können. Außerdem ist es eventuell notwendig, Informationen über den Swipe auszuwerten, wie z. B. Länge oder Richtung der Bewegung, um diese Werte mit vorgegebenen Einschränkungen wie Mindestlänge oder Bewegungsrichtung zu vergleichen. Dadurch kann entschieden werden, ob die Geste in einem bestimmten Kontext korrekt ausgeführt wurde oder nicht.

Zunächst wird die Modellierung *einfacher Gesten*<sup>9</sup> vorgestellt. Im nächsten Kapitel werden darauf aufbauend sog. synchrone Gesten eingeführt und das Modell dahin gehend erweitert, dass auch diese Gesten beschrieben werden können.



**Abbildung 3.6.:** Modell zur Erkennung einfacher Gesten.

Das Modell in Abbildung 3.6 ermöglicht es, den Prozess der Erkennung und Einschränkung einfacher Gesten zu beschreiben. Im ersten Schritt (*Recognize Gesture*) erfolgt die Verarbeitung von Sensor- und Eingabedaten, um das grundlegende Muster der Geste zu erkennen. Im zweiten Schritt (*Check Constraints*) wird geprüft, ob die Geste vorgegebenen Einschränkungen (*Constraints*) entspricht. Dementsprechend ist das Ergebnis des ersten Schrittes ein Event, das die Parameter der erkannten Geste beinhaltet (*GestureEvent*), welche im zweiten Schritt mit Einschränkung

<sup>9</sup>Einfache Gesten werden im Rahmen dieser Arbeit diejenigen Gesten genannt, die auf einem einzelnen Gerät erkannt werden, ohne dass eine Kommunikation mit anderen Geräten stattfindet.

gen an das Grundmuster, also den Anforderungen an ein erfolgreiches Erkennen der Geste, verglichen werden. Die abschließende Übergabe des *GestureEvents* an die weitere Applikationslogik ermöglicht die Steuerung der ausgelösten Systemreaktion oder die Erkennung synchroner Gesten (siehe Kapitel 3.3.3).

Die in [Hinckley u. a., 2004] beschriebene Stitch-Geste zerfällt zunächst in zwei Swipe-Gesten, die auf zwei Geräten unabhängig voneinander erkannt werden müssen und mit dem Modell aus Abbildung 3.6 erfasst werden können. Wie in Abbildung 3.4 zu sehen, muss dazu auf einem Gerät ein Swipe in Richtung des Bildschirmrandes erkannt werden und auf dem anderen Gerät ein Swipe vom Rand des Gerätes ausgehend zur Mitte des Bildschirms. Hinckley definiert zudem, dass ein Swipe auf dem ersten Gerät länger als 250 ms, auf dem zweiten Gerät länger als 100 ms dauern muss und dass eine nicht näher spezifizierte Mindestgeschwindigkeit vorausgesetzt wird, damit die Swipe-Gesten erkannt werden.

In Abbildung 3.7 ist die Modellierung des ersten Schritts (*Recognize Gesture*) für den Swipe auf einem Gerät exemplarisch vorgenommen worden. Da bei der Erkennung von Gesten auf Sensor- oder andere Eingabe-Events reagiert wird, wird zur Modellierung dieses Schritts ein Zustandsdiagramm verwendet. Benötigte Parameter werden mit *[params]* gekennzeichnet, benötigte Sensoren oder andere Datenquellen mit *[required]*. Der Austrittspunkt ist mit der anschließenden Aktivität (hier *Check Constraints: Swipe*) verknüpft, zurückgegebene Daten werden mit *[return]* gekennzeichnet.

Angelehnt an [Hinckley u. a., 2004] wird mit *min\_velocity* ein Schwellwert für die Geschwindigkeit als Parameter definiert, außerdem wird ein Touchscreen vorausgesetzt, der mindestens die folgenden Events unterscheiden kann:

- *TOUCH\_DOWN*: erste Berührung des Bildschirms.
- *TOUCH\_MOVE*: anhaltende Berührung des Bildschirms.
- *TOUCH\_UP*: letzte Berührung des Bildschirms.

Wurde ein *TOUCH\_DOWN* erkannt, werden Zeitpunkt und Koordinaten des Berührungspunktes gespeichert und auf ein *TOUCH\_UP* gewartet, eventuelle *TOUCH\_MOVE*-Events werden ignoriert. Aus den Start- und Endberührungen kann eine Geschwindigkeit errechnet werden, die mit dem Schwellwert *min\_velocity* verglichen wird. Liegt die Geschwindigkeit über dem Schwellwert, wird ein *SwipeEvent* generiert und an den zweiten Schritt, *Check Constraints* (siehe Abbildung 3.6), übergeben. Ansonsten beginnt die Erkennung von vorne.

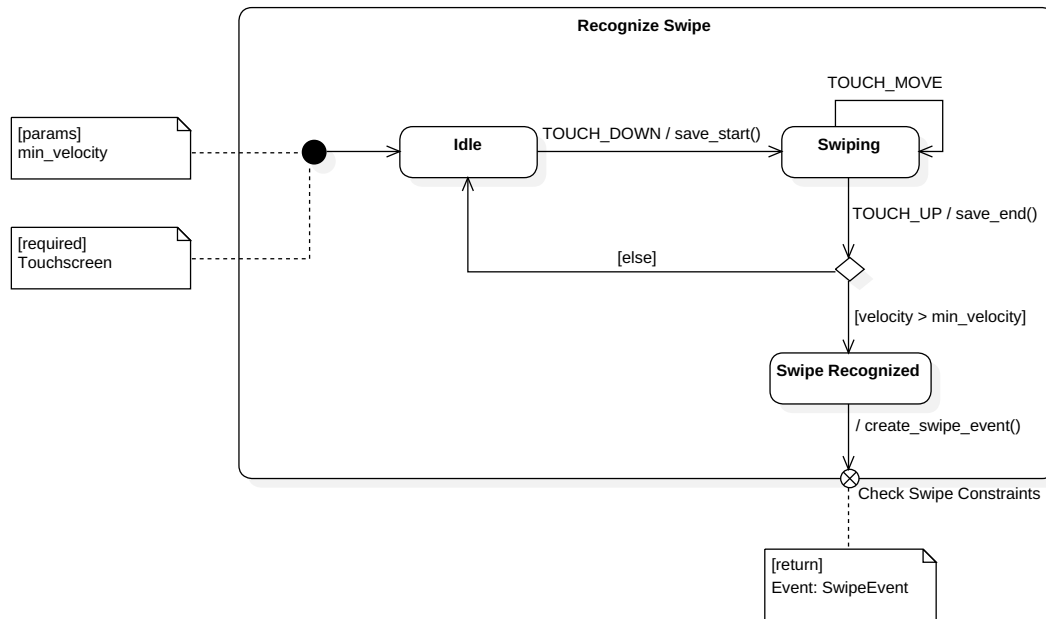


Abbildung 3.7.: Algorithmus zur Erkennung einer Swipe-Geste anhand von Touch-Events.

Die Modellierung des zweiten Schritts (*Check Constraints*) wurde in Abbildung 3.8 exemplarisch vorgenommen. Hierzu wird ein Aktivitätsdiagramm verwendet, da nicht auf Events reagiert wird, sondern nur prozedural geprüft wird, ob die Parameter des eingehenden Events (hier *SwipeEvent*) vordefinierten Einschränkungen (*Constraints*) entsprechen. In diesem Fall werden analog zu [Hinckley u. a., 2004] Richtung und zeitliche Dauer des Swipes überprüft. Diese Werte können anhand der Koordinaten und Zeitstempel der ersten und letzten Berührung des Swipes errechnet und mit den erwarteten Werten verglichen werden. Die Definition der Berechnungen findet sich als ergänzende Anmerkung im Diagramm. In diesem Beispiel wird der Swipe nur dann erkannt, wenn er mindestens 250 ms gedauert hat und in Richtung des rechten Bildschirmrandes durchgeführt wurde.

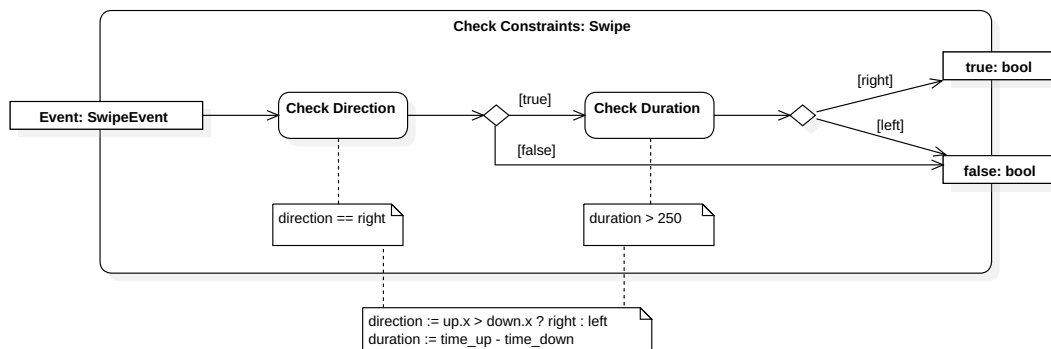
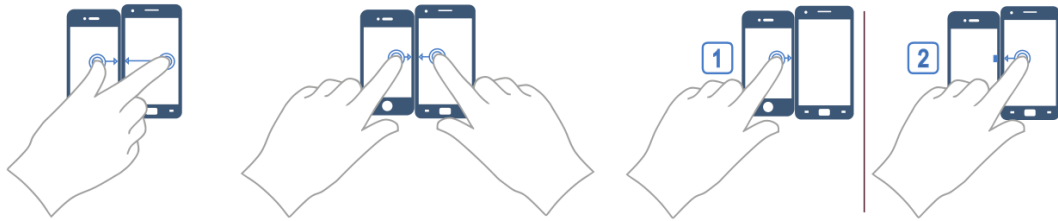


Abbildung 3.8.: Vergleich der erkannten Swipe-Geste mit vordefinierten Einschränkungen (*Constraints*).



**Abbildung 3.9.:** Drei Ausprägungen der Pinch-Geste. Quelle: [Nielsen u. a., 2014].

Ist die Aktivität *Check Constraints* erfolgreich, also der Rückgabewert *true*, so reicht die *Gesture Detection* aus 3.6 das *SwipeEvent* an die Applikation weiter, die Geste wurde erfolgreich erkannt. Ist die Prüfung nicht erfolgreich, wird die Erkennung erneut gestartet.

Durch Abwandeln oder Hinzufügen von Parametern für die *Gesture Recognition* und der Definition von *Constraint Checks* lassen sich so eine Vielzahl von *Swipe*-Gesten einfach beschreiben, wie z. B. Teile der *Pinch*-Geste, die in [Lucero u. a., 2011], [Chen u. a., 2014] oder [Nielsen u. a., 2014] verwendet wird (siehe Abbildung 3.9).

Auch die Erkennung von Gesten, die auf anderen Sensoren basieren, können mit dem Modell beschrieben werden. In [Lucero u. a., 2011] werden Fotos in einem virtuellen Fotostapel durch eine *Tilt*-Geste durchgeschaltet, indem das Kippen des Gerätes, das vorher ruhig auf einem Tisch lag, erkannt wird. Dazu wird ein Beschleunigungssensor eingesetzt, der bei ruhiger Lage auf dem Tisch nur die Erdbeschleunigung auf der Z-Achse misst (siehe Kapitel 2.4.2). Wird das Gerät nun kippend angehoben, verlagert sich die Beschleunigung auch auf die anderen beiden Achsen, wodurch die Kippbewegung des Gerätes festgestellt werden kann.

Im ersten Schritt (*Recognize Gesture*) wird zunächst das eigentliche Kippen, also der Übergang vom liegenden in den gekippten Zustand, festgestellt. Dazu wird ein Beschleunigungssensor benötigt, der als *required* angegeben wird. Zudem müssen eventuelle Schwankungen (z. B. durch leichte Erschütterungen oder leichte Schiefelage des Tisches) durch Schwellwerte berücksichtigt werden [Dachselt und Buchholz, 2009], die in den *params* definiert werden.

Anschließend kann das resultierende *GestureEvent* (in diesem Fall ein *TiltEvent*, das z. B. die Neigungswinkel beinhaltet) noch gegen *Constraints* geprüft werden, da in [Lucero u. a., 2011] z. B. nur horizontales Kippen zum Weiterschalten der Fotos verwendet wird. Das *TiltEvent* wird anschließend an die Applikationslogik übergeben, da es die folgende Systemreaktion eventuell beeinflusst. So schlagen

Cho und Kollegen z. B. vor, anhand des Neigungswinkels zu entscheiden, wie viele Fotos auf einmal weiter geschaltet werden sollen [Cho u. a., 2007]. Das Ergebnis ließe sich auch einfach auf [Rekimoto, 1996] oder [Dachselt und Buchholz, 2009] übertragen, da beide ebenfalls mit Tilt-Gesten arbeiten.

#### 3.3.3. Erkennung synchroner Gesten

Im vorigen Kapitel wurde die Modellierung einfacher Gesten beschrieben, deren Erkennung auf einem einzelnen Gerät stattfindet. Viele Multiscreen-Szenarien beinhalten allerdings Gesten, deren Durchführung mehrere Geräte und Nutzer gleichzeitig einbezieht. Hinckley hat dafür den Begriff der *synchronen Gesten* (engl. *synchronous gestures*) eingeführt, den er wie folgt definiert [Hinckley, 2003]:

These are patterns of activity, contributed by multiple users (or one user with multiple devices), which take on a new meaning when they occur together in time, or in a specific sequence in time. [...] These patterns could literally occur in parallel and in exact synchrony, or they just might be partially overlapped or even occur in a particular sequence. The key is that complementary portions of a signal are contributed by different devices or participants, and that the signal can only be recognized when these portions are brought together.

Viele Gesten, wie z. B. Bump [Hinckley, 2003], Stitch [Hinckley u. a., 2004], Pinch (u.a. verwendet in [Lucero u. a., 2011], [Chen u. a., 2014] und [Nielsen u. a., 2014]) oder das in [Rekimoto, 2004] vorgestellte System SyncTap fallen unter diese Definition der synchronen Gesten.

All diese Gesten haben die Gemeinsamkeit, dass zwar auch eine einfache Erkennung pro teilnehmendem Gerät stattfindet, die Erkennung der gesamten synchronen Geste aber zudem einen Abgleich der einzelnen *GestureEvents* und damit auch eine Kommunikation zwischen den einzelnen Geräten erfordert. Abbildung 3.10 zeigt eine Erweiterung des Modells, um synchrone Gesten beschreiben zu können. Zur Darstellung wird ein Sequenzdiagramm verwendet, das üblicherweise zur Visualisierung von Protokollen und Kommunikationsabläufen verwendet wird und hier den zeitlichen Ablauf ausgetauschter Nachrichten anschaulich abbildet.

Zwei Geräte (bezeichnet als Gerät A und B) verbinden sich mit einem externen Server, der im Netzwerk erreichbar ist, Nachrichten verschicken und empfangen oder Programmlogik ausführen kann. Auf beiden Geräten wird zunächst eine einfa-

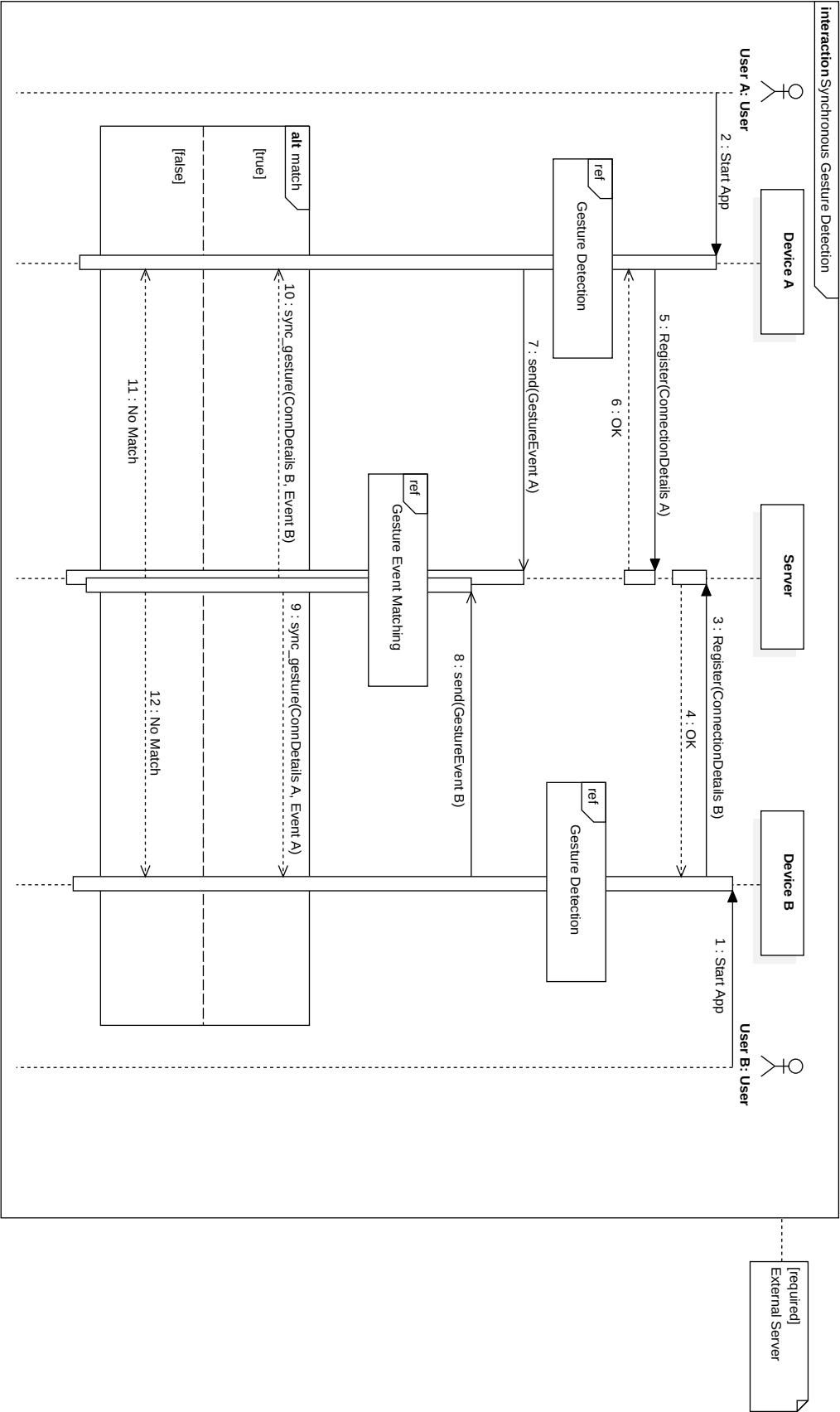
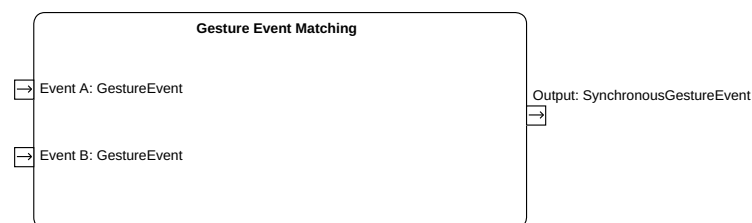


Abbildung 3.10.: Kommunikationsprotokoll zur Erkennung synchroner Gesten.

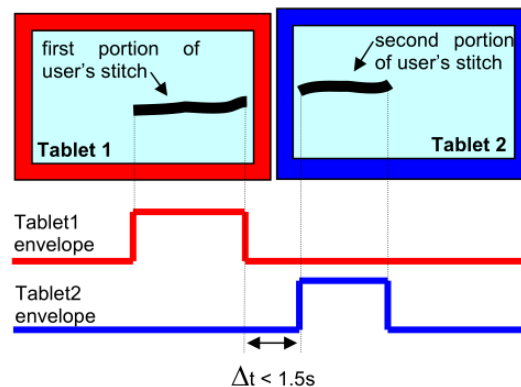


che Gestenerkennung durchgeführt, hier als Referenz auf die in Kapitel 3.3.2 beschriebene einfache Gestenerkennung. Die resultierenden *GestureEvents* werden anschließend von beiden Geräten an den Server gesendet, der einen Vergleich aller eingehenden Events durchführt. Wird ein erfolgreicher Vergleich zweier Events vorgenommen, also eine synchrone Geste erkannt, informiert der Server die beteiligten Geräte über die Verbindungsparameter (*ConnectionDetails*) ihrer jeweiligen Kommunikationspartner.

Abbildung 3.11 zeigt eine Blackbox-Sicht auf die Aktivität *Gesture Event Matching*, welche den Vergleich von zwei *GestureEvents* vornimmt. Zwei beliebige Events, *Event A* und *Event B*, werden als Eingabe akzeptiert. Innerhalb der Aktivität wird ein Vergleich der Event-Parameter durchgeführt, der analog zur einfachen Gestenerkennung einen Wahrheitswert zurückgibt. Besteht zwischen den beiden Events ein Zusammenhang, bilden sie zusammen eine synchrone Geste.



**Abbildung 3.11.:** Vergleich von *GestureEvents* zur Erkennung synchroner Gesten.



**Abbildung 3.12.:** Zusammenhang zwischen Swipe-Events und Stitch-Erkennung. Quelle: [Hinckley u. a., 2004].

Ein detailliertes Beispiel für die Erkennung einer synchronen Geste liefern Hinckley und Kollegen mit der Beschreibung der Stitch-Geste (s. auch Kapitel 3.3.1) [Hinckley u. a., 2004]. Zunächst wird pro Gerät ein Swipe erkannt, der gewissen Constraints unterliegt. So müssen die Swipes eine zeitliche Mindestlänge und eine

### 3. Modell für Multiscreen-Interaktionen

bestimmte Richtung aufweisen. Bis dahin handelt es sich um einfache Gesten. Eine exemplarische Beschreibung solcher Swipe-Gesten durch das Modell wurde bereits in Kapitel 3.3.2 vorgenommen. Anschließend wird ein Vergleich definiert, ob die jeweiligen Swipes (vgl. Abbildung 3.12)

- eine durchgehende Linie bilden, also ob beide Swipe-Gesten die gleiche Richtung haben,
- beide Swipe-Gesten mit einem zeitlichen Abstand von  $t < 1.5\text{ s}$  ausgeführt wurden und
- der Austrittswinkel  $\alpha_1$  auf Tablet 1 gleich dem Eintrittswinkel  $\alpha_2$  auf Tablet 2 ist, mit einer Toleranz von  $\pm 20^\circ$ .

Dazu wird im Modell ein *Stitch Event Matching* definiert, das zwei *SwipeEvents* als Eingabeparameter hat und einen Wahrheitswert zurückgibt, ob eine synchrone Geste erkannt wurde. Abbildung 3.13 zeigt die Modellierung dieses Vergleichs. Ein Aktivitätsdiagramm beschreibt die Abfolge der Prüfungen. Durch Annotationen können einfache Vergleiche oder Berechnungen komplexerer Parameter auf Basis der *SwipeEvent*-Parameter definiert werden. In diesem Fall werden die oben beschriebenen Werte Richtung, Zeitdifferenz und Ein- bzw. Austrittswinkel der Swipes berücksichtigt. Setzt man in dem Modell in Abbildung 3.10 für die *Gesture Detection* eine Swipe-Erkennung wie in Kapitel 3.3.2 und für das *Gesture Event Matching* das Stitch Event Matching ein, erhält man eine vollständig beschriebene, synchrone Geste.

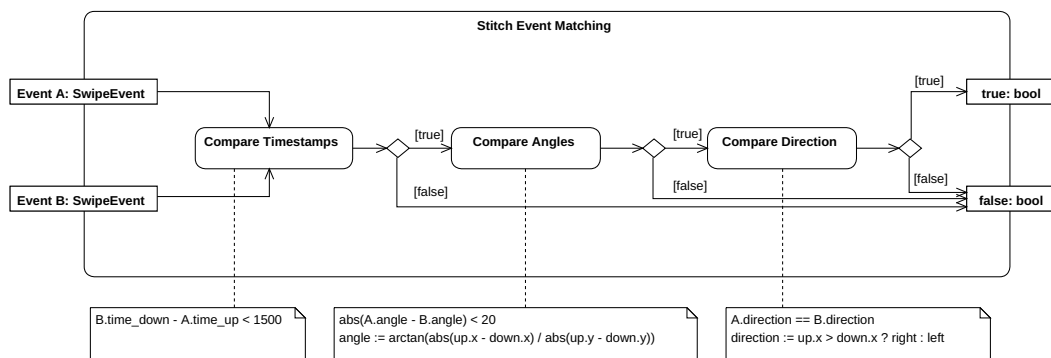
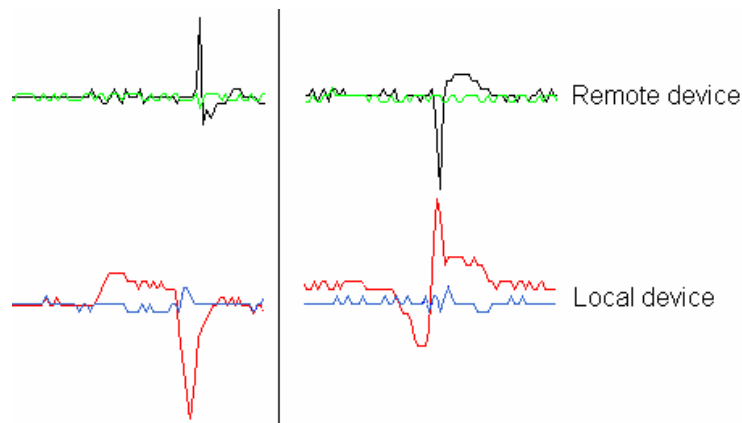


Abbildung 3.13.: Vergleich zweier Swipe-Events zur Erkennung der synchronen Stitch-Geste.

Hinckley stellt mit der Bump-Geste eine weitere synchrone Geste im Detail vor [Hinckley, 2003]. Zwei Geräte (bezeichnet als *lokales* und *entferntes* Gerät), die über einen Beschleunigungssensor verfügen, überwachen dabei kontinuierlich ihre

Beschleunigung. Ein Bump wird zunächst je Gerät über eine Spitze (engl. *peak*) in der Beschleunigungskurve erkannt (siehe Abbildung 3.14). Eine Spitze liegt vor, wenn eine vordefinierte Schwelle über- und anschließend wieder unterschritten wird, sofern dies in einem Zeitfenster von maximal 65 ms passiert. Im resultierenden *BumpEvent* wird der Zeitpunkt sowie die Richtung der maximalen Beschleunigung innerhalb der Spitze notiert. Dieser Vorgang kann soweit als einfache Geste beschrieben werden (vgl. Kapitel 3.3.2). Anschließend wird das lokale *BumpEvent* mit denen eines entfernten Gerätes verglichen, zu dem eine Netzwerkverbindung besteht. Gab es auf dem entfernten Gerät eine Spitze, deren Zeitstempel maximal 50 ms von der lokalen Spitze entfernt und deren Richtung entgegengesetzt ist (vgl. Abbildung 3.14), wird die synchrone Geste erkannt und beide Geräte über ihren Kommunikationspartner informiert.



**Abbildung 3.14.:** Typische Beschleunigungsmuster der Bump-Geste. Quelle: [Hinckley, 2003].

Auch die Bump-Geste lässt sich analog zum Stitch durch das Modell beschreiben. Es müssen entsprechend dem Modell in Abbildung 3.10 folgende Komponenten spezifiziert werden:

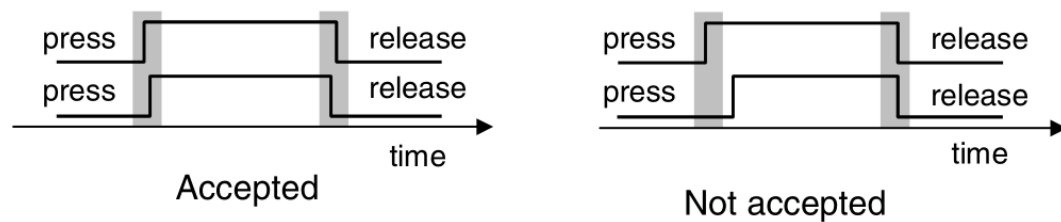
- Eine einfache *Bump Detection*, die Werte eines Beschleunigungssensors pro Gerät überwacht, auf Spitzen reagiert und ein *BumpEvent* erzeugt.
- Ein *Bump Event Matching*, das je zwei *BumpEvents* entgegennimmt, Richtung und Zeitpunkt der Maximalbeschleunigungen vergleicht und einen Wahrheitswert zurückgibt, ob eine synchrone Geste vorliegt oder nicht.

Ein weiteres Beispiel für synchrone Gesten findet sich in [Rekimoto, 2004]. Hier wird das synchrone Drücken und Loslassen von Buttons erkannt. Wurde auf beiden Geräten ein Knopf gleich lange gedrückt (mit einer Toleranz  $c_1 = 100$  ms) und zum gleichen Zeitpunkt wieder losgelassen (mit einer Toleranz  $c_2 = 200$  ms), wird eine

### 3. Modell für Multiscreen-Interaktionen

synchrone Geste erkannt (siehe Abbildung 3.15). Spezifiziert werden müssen auch hier:

- Eine *Gesture Detection*, die *ButtonEvents* erzeugt.
- Ein *Gesture Event Matching*, das Länge und Zeitpunkte der *ButtonEvents* unter Einbeziehung von  $c_1$  und  $c_2$  vergleicht.



**Abbildung 3.15.:** Erkennung einer synchronen Geste basierend auf Buttons im System SyncTap. Quelle: [Rekimoto, 2004].

Eine Ergänzung muss an dieser Stelle noch vorgenommen werden. Das Modell in Abbildung 3.10 geht vom einfachsten Fall aus, in dem ein Server zwischen den einzelnen Geräten vermittelt. Dadurch können auch Geräte miteinander verbunden werden, die sich vorher nicht kennen. In [Hinckley u. a., 2004] wird eine Infrastruktur vorgestellt, in der ein Gerät selbst die Rolle des Servers übernimmt und dessen Adresse allen anderen Geräten bekannt sein muss. In dem Fall wird keine zusätzliche Vermittlungskomponente benötigt, ebenso wenn zwei Geräte bereits per Ad-hoc-Netzwerk miteinander verbunden sind und somit unmittelbar eine synchrone Gestenerkennung durchführen können.

In [Rekimoto, 2004] wird zudem eine Alternative beschrieben, in der sich alle Geräte in einem gemeinsamen LAN bzw. WLAN befinden und UDP<sup>10</sup>-Pakete mit Button-Events an alle SyncTap-Netzwerkteilnehmer versenden (Multicast). Das *Gesture Event Matching* wird in dem Fall auf jedem Gerät parallel ausgeführt, anstatt nur auf einem dedizierten Gerät. Für beide Fälle muss das Modell aus 3.10 leicht abgewandelt werden, die Ergänzungen befinden sich in Anhang C.

Im folgenden Kapitel wird beschrieben, wie der Verbindungsaufbau zwischen Geräten im Allgemeinen gestaltet werden und wie die Erkennung einfacher oder synchroner Gesten in diesen Ablauf eingebettet werden kann.

<sup>10</sup>Das User Datagram Protocol (UDP) ist ein simples, verbindungsloses Netzwerkprotokoll, um Pakete in einem Netzwerk zu versenden und zu empfangen. UDP wird oft eingesetzt, wenn Paketverluste akzeptabel sind oder die Reihenfolge des Eintreffens von gesendeten Paketen keine Rolle spielt.

#### 3.3.4. Connect

Das Rahmenmodell in Kapitel 3.3.1 sieht im Lebenszyklus von Multiscreen-Anwendungen zunächst das Verbinden zweier Geräte in Form der Aktivität *Connect* vor, deren Details in diesem Kapitel beschrieben werden.

In der SysPlace-Patternsammlung finden sich unter der Kategorie *Connect* verschiedene Patterns zum Aufbauen einer Verbindung zwischen zwei Geräten, die alle auf einer (zumeist synchronen) Geste basieren, wie z. B. der Bump- und Stitch-Geste oder dem gleichzeitigen Schütteln zweier Geräte. Wurde die synchrone Geste erkannt, wird auf beiden Geräten gleichzeitig eine Systemreaktion ausgelöst, die den Verbindungsaufbau zwischen diesen Geräten zum Ziel hat. Abbildung 3.16 zeigt im Detail, welche Aktivitäten und System-Feedbacks dabei auf beiden Geräten ausgeführt werden und wie sie in Beziehung zueinander stehen. Dazu wird ein Aktivitätsdiagramm verwendet, das zwischen Aktivitäten (pro Gerät) und Events unterscheidet, die entweder die Kommunikation zwischen den Geräten bzw. eventuelle Fehlerfälle darstellen.

Angestoßen wird die *Connect*-Aktivität, wenn eine synchrone Geste erkannt wurde. Kapitel 3.3.2 und 3.3.3 beschreiben die Modellierung der Gestenerkennung ausführlich, in diesem Diagramm wird lediglich eine Blackbox-Sicht verwendet. Dadurch wird die Geste komplett austauschbar, da für den weiteren Verbindungsaufbau lediglich die Parameter für die Verbindung, wie z. B. Geräteadressen, in Form von *ConnectionDetails* bekannt sein müssen.

Sobald die Kommunikationspartner über die *ConnectionDetails* informiert sind, akzeptiert eines der Geräte eingehende Verbindungen, während das andere Gerät versucht, sich mit diesem zu verbinden. Kann die Verbindung erfolgreich hergestellt werden, ist das Ergebnis ein bidirektionaler Datenkanal (*Channel*), über den beide Geräte Nachrichten austauschen können, die Aktivität *Connect* wird dann erfolgreich beendet. Kommt es zu einem Fehler beim Verbindungsaufbau, endet die gesamte Aktivität auf beiden Geräten ohne Verbindung.

Die Nutzer beider Geräte erhalten an verschiedenen Stellen jeweils ein *Feedback* des Systems, ob die Verbindung aufgebaut werden konnte oder nicht (siehe orangene Aktivitäten in Abbildung 3.16, die mit *Feedback* gekennzeichnet sind). Solche Feedback-Aktivitäten sind lediglich Design-Hinweise, dass es an dieser Stelle sinnvoll sein kann, den Nutzer über den Systemzustand zu informieren. Wie die Rückmeldung gestaltet wird, muss für jede Anwendung individuell entschie-

### 3. Modell für Multiscreen-Interaktionen

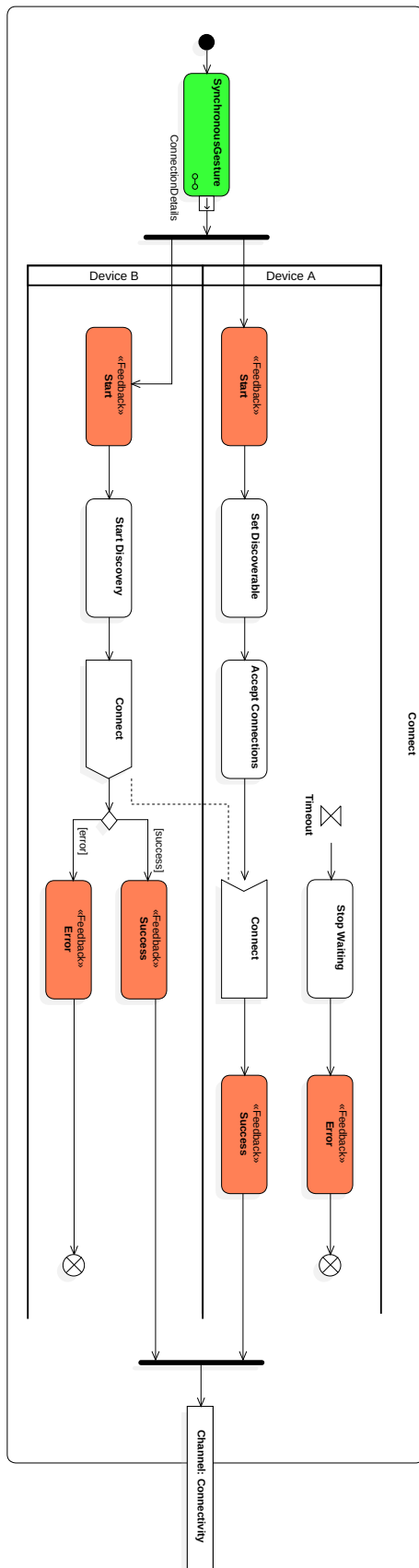


Abbildung 3. 16.: Aufbauen einer Verbindung zwischen zwei Geräten mittels synchroner Geste.

den werden. Gestaltungsmöglichkeiten entsprechend den Ausgabetechnologien aus Kapitel 2.4.3 sind u.a.:

- *visuelles Feedback*: Veränderungen in der Nutzeroberfläche, wie z. B. Einblendung von Textboxen, Animationen oder Ändern der Farbe.
- *akustisches Feedback*: Abspielen von Geräuschen, um den Nutzer über Veränderungen zu informieren und evt. visuelles Feedback zu unterstützen.
- *Vibration*: Mobile Geräte können Vibrationen erzeugen, um den Anwender über Änderungen zu informieren.

Da System-Feedback beliebig gestaltet und kombiniert werden kann, ist eine modellhafte Erfassung an dieser Stelle nur eingeschränkt sinnvoll. Es ist lediglich möglich, die benötigten Ausgabetechnologien zu erfassen, die für eine bestimmte Ausgestaltung des Feedbacks mindestens benötigt werden. Dazu wird in Kapitel 3.3.8 beschrieben, wie System-Feedback unter Einbeziehung benötigter Ausgabetechnologien statisch durch das Modell erfasst werden kann.

In der Literatur finden sich vielerlei Hinweise für die Gestaltung des Verbindungsprozesses sowie des Feedbacks. Einige Beispiele, die bei der Erstellung des Modells berücksichtigt wurden, werden im Folgenden dargestellt.

Hinckley schlägt für die Verwendung der Bump-Geste zum Verbindungsaufbau verschiedene Feedback-Mechanismen vor [Hinckley, 2003]:

- Ein kurzes, metallisches Klicken auf dem Tablet, welches die Geste erkannt hat, um ein „Zusammenschnappen“ der Tablets anzudeuten (*akustisches Feedback*).
- Einen Pfeil zum Bildschirmrand, der andeutet, dass eine Verbindung mit einem anderen Tablet aufgebaut wird (*visuelles Feedback*).
- Einen „popping sound“, der auf dem zweiten Tablet abgespielt wird, wenn der Verbindungsaufbau erfolgreich abgeschlossen wurde (*akustisches Feedback*).
- Einen zweiten, kleineren Pfeil zum Bildschirmrand des zweiten Tablets, der mit dem ersten Pfeil zusammen nach zwei Sekunden wieder ausgeblendet wird (*visuelles Feedback*).

Hinckley betont, dass es wichtig sei, visuelles und akustisches Feedback zu kombinieren, falls ein Nutzer z. B. gerade nicht seinen Fokus auf den Tablets hat und eine Verbindung zustande gekommen ist.

Ferner merkt er an, dass das Feedback auf beiden Geräten zwar oft simultan und redundant stattfindet, es aber dennoch in jedem Fall stattfinden sollte, falls der Verbindungsaufbau bspw. länger dauert und der Nutzer sonst nicht weiß, ab wann eine Verbindung besteht. Die Geräte werden über WLAN miteinander verbunden, als Netzwerkprotokoll wird TCP<sup>11</sup> eingesetzt [Hinckley, 2003].

Das Modell in Abbildung 3.16 ermöglicht es, diese Design-Hinweise zu berücksichtigen. So können das akustische Feedback (metallisches Klicken) sowie der visuelle Hinweis (der Pfeil) als *Feedback Start* auf dem Tablet eingeblendet werden, das auf eingehende Verbindungen wartet. Sobald das zweite Tablet sich verbunden hat, kann auf diesem der zweite Sound sowie der entgegengesetzte Pfeil als *Feedback Success* eingeblendet werden. Zudem werden nach zwei Sekunden beide Pfeile ausgeblendet, was ebenfalls unter *Feedback Success* auf beiden Tablets berücksichtigt werden muss. Hinckley definiert für den Fall, dass keine Verbindung zustande kommt, kein explizites Feedback [Hinckley, 2003]. Auf dem Tablet, das auf eingehende Verbindungen wartet, muss dann zwar als *Feedback Error* der Pfeil ausgeblendet werden, für das andere Tablet würde aber kein *Feedback Error* definiert.

Auch das verwendete Netzwerkprotokoll TCP findet sich vereinfacht in dem Modell wieder (Akzeptieren von Verbindungen auf Gerät A bzw. aktives Verbinden von Gerät B zu Gerät A). Weitere Details werden hier bewusst ausgelassen, da der Verbindungsaufbau der meisten Netzwerktechnologien nach einem ähnlichen Schema funktioniert und eine detaillierte Beschreibung das Modell unnötig verkomplizieren würde.

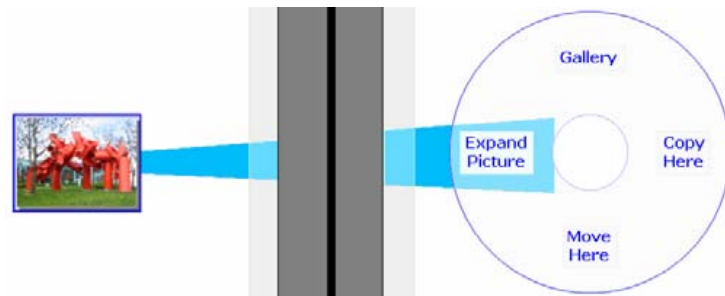
Für die Stitch-Geste finden sich ebenfalls Designhinweise bzgl. der *Connect*-Aktivität, die sich durch das Modell beschreiben lassen [Hinckley u. a., 2004]. So definieren Hinckley und Kollegen, dass ein kurzer Piepton abgespielt wird, sobald die Stitch-Geste auf dem zweiten Tablet erkannt wird (siehe Abbildung 3.12). Zudem können eventuell zu übertragene Daten auf dem ersten Tablet visuell hervorgehoben werden. Hierfür wird keine bestehende Verbindung benötigt, beides kann als *Feedback Start* definiert werden. Hinckley beschreibt zudem, dass bei erfolgreicher Verbindung ein Menü eingeblendet wird, welches dem Nutzer Operationen auf den selektierten Daten vom ersten Tablet anbietet (z. B. Kopieren der Daten auf das zweite Tablet). Das ist erst sinnvoll, wenn bereits eine Verbindung besteht, also in

---

<sup>11</sup>Das Transmission Control Protocol (TCP) ist ein verbindungsorientiertes Netzwerkprotokoll, in dem im Gegensatz zu UDP verlorene Pakete neu gesendet werden und die Reihenfolge gesendeter Pakete beachtet wird.



der Aktivität *Feedback Success*. Das Menü geht über beide Bildschirme (siehe Abbildung 3.17) und muss dementsprechend für beide Geräte definiert werden.



**Abbildung 3.17.:** Optionsmenü als visuelles Feedback nach erfolgreicher Stitch-Geste. Quelle: [Hinckley u. a., 2004].

Anzumerken ist, dass beim Stitch meist die Rahmenmodell-Variante aus Abbildung 3.5 zum Einsatz kommt, da teilweise vor dem Verbinden von Geräten schon Daten selektiert werden. Als Netzwerktechnologie kommt WLAN zum Einsatz. Es wird kein Netzwerkprotokoll explizit erwähnt, vermutlich wird aber analog zur Bump-Geste in [Hinckley, 2003] wieder TCP verwendet.

Zwar werden meist synchrone Gesten zum Aufbauen einer Verbindung verwendet, allerdings sind auch einfache Gesten auf einem Gerät denkbar, um den Verbindungsaufbau einseitig anzustoßen. Abbildung 3.18 zeigt eine Abwandlung des Modells, das auf einfachen Gesten basiert. In diesem Fall führt nur der Nutzer von Gerät A eine Geste aus, durch die ein Verbindungsaufbau zu Gerät B initiiert wird. Der Nutzer von Gerät B muss die Möglichkeit haben, die Verbindung entweder anzunehmen oder abzulehnen, was durch eine Aktivität vom Typ *User Interaction* berücksichtigt wurde, die analog zum Feedback individuell an die Applikation angepasst werden kann.

Eine interessante Anwendung dazu findet sich im System BlueTable [Wilson und Sarin, 2007]. Hier soll die exakte Position von mobilen, Bluetooth-fähigen Geräten auf einer interaktiven Oberfläche<sup>12</sup> erkannt werden, ohne auf Technologien wie RFID oder NFC zurückzugreifen. Dazu schlagen die Autoren folgenden Ablauf vor:

1. Die interaktive Oberfläche erkennt, dass ein Smartphone-förmiges Objekt an einer bestimmten Position aufgelegt wurde.

<sup>12</sup>Mit interaktiver Oberfläche ist hier ein System gemeint, das Berührungen oder physikalische Objekte auf einer Fläche erkennen und tracken kann. In [Wilson und Sarin, 2007] wird dazu das System Play-Anywhere verwendet, das aus einem Projektor und einer Kamera besteht und normale Oberflächen digital erfassen kann (eine ausführliche Beschreibung von PlayAnywhere findet sich in [Wilson, 2005]).

### 3. Modell für Multiscreen-Interaktionen

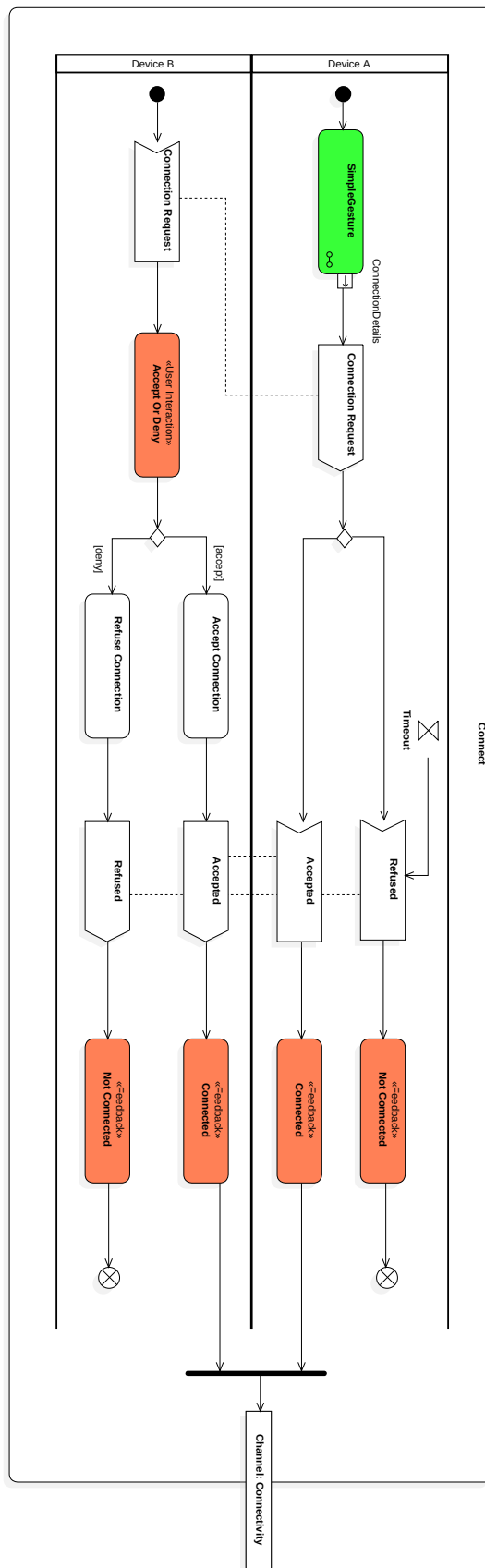


Abbildung 3.18.: Aufbauen einer Verbindung zwischen zwei Geräten mittels einfacher Geste.

2. Das System baut zu jedem Bluetooth-fähigen Gerät in Reichweite eine Verbindung auf, sofern dieses eine vordefinierte BlueTable-Service-ID aussendet.
3. Sobald eine Verbindung besteht, wird das Gerät aufgefordert, seinen Infrarot-Port aufblinken zu lassen.
4. Stimmt die Position des Blinkens mit der Position des Auflegens überein, besteht eine Bluetooth-Verbindung zu dem Smartphone an exakt dieser Position.

Definiert man das Auflegen eines Smartphones als Geste auf der interaktiven Oberfläche, kann man dieses System durch das Modell in Abbildung 3.18 beschreiben. Gerät A (die interaktive Oberfläche) erkennt das Auflegen eines Gerätes B und sendet eine Bluetooth-Anfrage an alle Bluetooth-Geräte in Reichweite. Erhält ein beliebiges Gerät diese Anfrage, kann die Verbindung akzeptiert oder abgelehnt werden. Dazu wird analog zu Feedback-Aktivitäten eine Aktivität vom Typ *User Interaction* eingeführt, die je nach Anwendungskontext individuell gestaltet werden muss. Die Option, die Verbindung abzulehnen, ist für alle Geräte sinnvoll, die außer Gerät B die Anfrage erhalten, weil sie zwar in Bluetooth-Reichweite sind, aber nicht aufgelegt wurden.

Akzeptiert ein Gerät die Verbindungsanfrage, ergibt sich hier als Besonderheit, dass das *Feedback Connected* neben eventuellen anderen Ausgaben das Blinken der Infrarot-Lampe beinhaltet und somit die Positionsbestimmung ermöglicht. Haben mehrere Geräte die Verbindung akzeptiert und ein Infrarot-Blinken ausgelöst, hält Gerät A die Verbindung nur zu demjenigen Gerät, dessen Position mit der Auflegeposition übereinstimmt, also Gerät B.

Dieser Bluetooth-Kanal wird anschließend für die weitere Applikationslogik genutzt, um z. B. Kontaktinformationen oder Bilder auszutauschen [Wilson und Sarin, 2007], was nach dem Rahmenmodell (siehe Kapitel 3.3.1) in den Aktivitäten *Select* bzw. *Transfer* passiert, welche in den folgenden Kapiteln näher beschrieben und modelliert werden.

#### 3.3.5. Select

Um Multiscreen-Szenarien zu realisieren, müssen Daten zwischen Geräten ausgetauscht werden. Diese werden entweder implizit durch den Kontext oder explizit

durch den Nutzer ausgewählt, bevor die Übertragung stattfinden kann. Das Rahmenmodell in Kapitel 3.3.1 sieht dazu die Aktivität *Select* vor, die entweder vor (siehe Abbildung 3.5) oder nach der *Connect*-Aktivität (siehe Abbildung 3.1) stattfindet, in jedem Fall aber dem *Transfer* von Daten vorausgeht. Auch eine wiederholte Ausführung der *Select*-Aktivität ist möglich, wenn mehrfach Daten übertragen werden sollen. Die Art der selektierten Daten entscheidet darüber, wie der anschließende Übertragungsprozess, also die Aktivität *Transfer*, gestaltet werden kann.

Da das Format von Daten und damit die Gestaltung des Selektionsprozesses stark vom jeweiligen Anwendungskontext abhängt, ist eine Modellierung der *Select*-Aktivität nur eingeschränkt möglich. Schwierig ist dabei die Erfassung des „Wie“, also in welcher Art und Weise die Interaktion des Nutzers mit dem System gestaltet wird. Zwar können hier auch Gesten zum Einsatz kommen, die mittels des Modells erfasst werden können (siehe Kapitel 3.3.2), für die Gestaltung der Oberflächen und der Interaktionsformen für die Selektion wird hier allerdings kein allgemeingültiges Modell aufgestellt. Das hat den Grund, dass Oberflächen stark an einen Anwendungskontext sowie die verwendete Plattform gebunden sind und die einheitliche, abstrakte Beschreibung von Nutzeroberflächen ein komplexes Thema ist, das im Rahmen dieser Arbeit nicht berücksichtigt werden kann. Erfasst werden kann aber das „Was“, also welcher Art die selektierten Daten sind. Abbildung 3.19 zeigt den Ablauf der Aktivität.

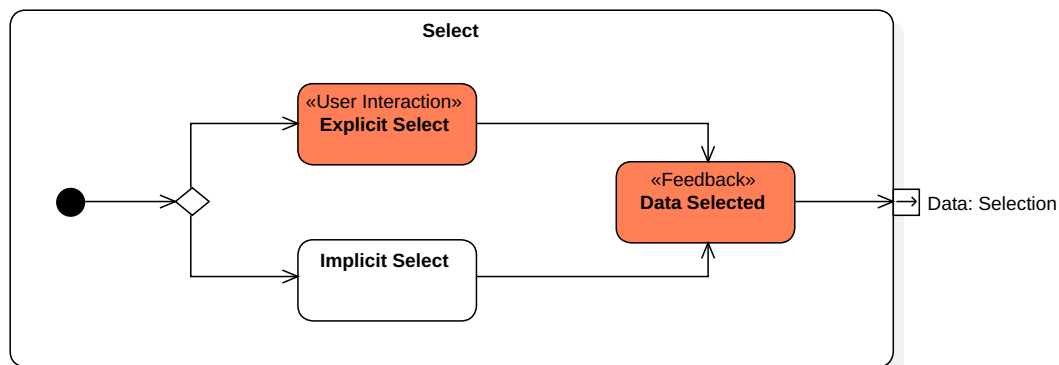


Abbildung 3.19.: Ablauf der Aktivität *Select*.

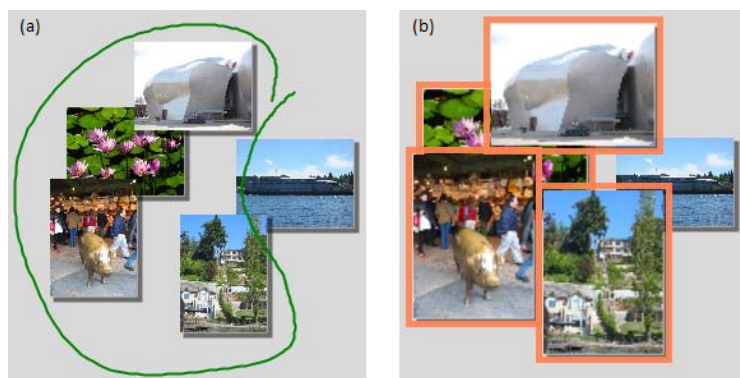
Hier sind die beiden Fälle der impliziten Auswahl durch die Anwendung bzw. der expliziten Auswahl durch den Benutzer (in Form einer Aktivität vom Typ *User Interaction*) berücksichtigt. Zudem kann ein zusätzliches System-Feedback ausgeführt werden, um den Nutzer über die erfolgte Auswahl zu informieren. Das Ergebnis der *Select*-Aktivität ist eine Auswahl (*Selection*) von Daten, deren Typ je nach

Kontext variieren kann. Für das Modell wurden vier verschiedene Typen herausgearbeitet:

- *File*: genau eine Datei, z. B. ein Foto oder Textdokument.
- *MultipleFiles*: eine Menge einzelner Dateien, z. B. alle Fotos in einem Ordner.
- *Stream*: ein Datenstrom (z. B. Videodaten), der kontinuierlich übertragen werden soll.
- *Command*: ein anwendungsspezifischer Steuerbefehl für die entfernte Anwendung.

Im System StitchMaster wird eine Selektion einzelner Dateien bzw. einer Menge von Dateien (meist Fotos) implementiert [Hinckley u. a., 2004]. Der Nutzer kann entweder ein einzelnes Foto (durch Antippen) oder eine Menge von Fotos (mittels einer Lasso-Geste) auswählen, was nach dem Modell in Abbildung 3.19 einer *expliziten Selektion* eines einzelnen Fotos (*File*) bzw. mehrerer Fotos (*MultipleFiles*) entspricht. Bei der Lasso-Geste wird ein Bereich mit Bildern markiert, dessen Inhalt dann die *Selection* bildet (siehe Abbildung 3.20 (a)). Die Beschreibungen der Gesten sind Teil der *User Interaction*.

Um den Benutzer zu informieren, welche Dateien zur Übertragung ausgewählt wurden, werden die Fotos anschließend mit einem Rahmen versehen (siehe Abbildung 3.20 (b)), was als visuelles Feedback *Data Selected* im Modell berücksichtigt werden kann. Als Ergebnis der Aktivität wird die *Selection* des entsprechenden Typs an die Transfer-Aktivität übergeben.



**Abbildung 3.20.:** Auswahl mehrerer Dateien im System StitchMaster. (a) Auswahl der Dateien mittels Lasso-Geste. (b) Hervorhebung der Auswahl durch das System. Quelle: [Hinckley u. a., 2004].

Auch in [Hinckley, 2003] werden verschiedene Gestaltungsmöglichkeiten des Selektionsvorgangs besprochen, die der Übertragung mittels Bump-Geste vorausgehen können. So wählt der Benutzer im einfachsten Fall explizit ein Foto (*File*) aus, das auf dem anderen Bildschirm angezeigt werden soll. Dabei wird nach der Übertragung entweder eine Kopie angezeigt, oder das Bild wird auf beide Bildschirme aufgeteilt (sog. *Display Tiling*). In diesem Fall müssen noch zusätzliche *Commands* übergeben werden, um z. B. die Bildschirmgröße zu übermitteln, damit beide Geräte ihren jeweiligen Teilausschnitt des Bildes berechnen können. Diese *Commands* werden implizit durch die Anwendung ermittelt, es findet keine weitere Nutzerinteraktion statt.

Auch eine sog. *Face-to-face Collaboration* wird beschrieben, bei der Benutzer ein geteiltes Whiteboard auf beiden Geräten benutzen können. In dem Fall wäre die *Selection* vom Typ *Stream*, da kontinuierlich Eingaben auf beiden Whiteboards synchronisiert werden müssen.

Ein weiterer Anwendungsfall ist das Teilen einer betrachteten Webseite. Ist auf einem Gerät eine Webseite geöffnet und es wird eine Bump-Geste ausgeführt, soll auf dem zweiten Gerät der Browser geöffnet und die gleiche Seite angezeigt werden [Hinckley, 2003]. Modelliert werden könnte dies mittels eines *Commands*, der als Parameter die zu startende Applikation (Browser) und den Startparameter (URL) übermittelt.

Analog dazu könnte der ebenfalls beschriebene Anwendungsfall erfasst werden, in der ein Word-Dokument geteilt und zur selben Stelle gesprungen werden soll [Hinckley, 2003]. Hier wären die Parameter die gleichen, nur die Werte für Applikation (Word) und Startparameter (Netzwerkpfad zum Dokument und Seitennummer) wären unterschiedlich. In beiden Fällen ist die Art der Selektion eher impliziter Natur, da der Nutzer nur die Anwendung in einem bestimmten Zustand geöffnet hat und diese automatisch die aktuelle Auswahl in Form eines *Commands* aus diesem Zustand ableitet.

Dachselt und Kollegen gehen ebenfalls auf den Auswahlprozess und die Art der ausgewählten Daten in ihrem Multiscreen-Szenario ein (siehe auch Kapitel 3.3.1 für eine kurze Beschreibung des Systems) [Dachselt und Buchholz, 2009]. Im ersten Schritt werden alle auf einem Smartphone vorhandenen Fotos implizit selektiert, die mittels der anschließenden Wurf-Geste übertragen werden können. Nach der Übertragung hat der Benutzer die Möglichkeit, die übertragenen Bilder auf dem

Smart TV durch Kippen des Smartphones durchzuschalten. Dabei wird mittels einer Tilt-Geste explizit ein *Command* selektiert, der dem Smart TV mitteilt, zum nächsten Foto zu schalten. Zudem können die Bilder durch eine weitere Geste vom Smart TV entfernt werden, was wiederum eine explizite *User Interaction* in Form einer Fetch-Back-Geste ist, aus welcher ein *Command* zum Entfernen der Bilder abgeleitet wird.

Im Sinne des Applikationslebenszyklus (siehe Kapitel 3.3.1) ergibt sich in dem Fall also eine mehrfache Ausführung der Aktivität *Select*, wobei je nach Zustand der Applikation entweder Dateien oder eine Folge von Steuerbefehlen übermittelt werden, um das gesamte Multiscreen-Szenario zu realisieren. Jede dieser Ausprägungen der *Select*-Aktivität muss modelliert werden, wenn die Anwendung komplett abgebildet werden soll.

Im folgenden Kapitel wird der Zusammenhang zwischen einer abgeschlossenen *Select*-Aktivität und dem Auslösen des Transfers beschrieben. Zudem wird ein für alle *Selection*-Typen einheitlicher Transfer-Prozess modelliert.

#### 3.3.6. Transfer

Entsprechend dem Rahmenmodell in Kapitel 3.3.1 findet die *Transfer*-Aktivität entweder unmittelbar nach den Aktivitäten *Connect* oder *Select* statt, in jedem Fall aber erst dann, wenn beide Aktivitäten einmal ausgeführt wurden und damit zwei Voraussetzungen erfüllt sind:

- Es besteht eine bidirektionale<sup>13</sup> Verbindung zwischen zwei Geräten (*Channel*).
- Auf einem Gerät wurden Daten zur Übertragung ausgewählt (*Selection*).

Abbildung 3.21 zeigt den Beginn des Transfer-Ablaufes unter Berücksichtigung dieser Voraussetzungen. Erst wenn beide Inputs vorhanden sind, kann ein Transfer durchgeführt werden. Ein Auslöser wird benötigt, um den Transfer der ausgewählten Daten zu starten. Hier gibt es zwei Möglichkeiten:

- Der Auslöser ist eine einfache oder synchrone Geste oder

---

<sup>13</sup>Der Einfachheit halber wird hier immer eine bidirektionale Verbindung angenommen, da die meisten Netzwerktechnologien bidirektional funktionieren und somit keine weitere Unterscheidung zwischen Aktivitäten mit uni- oder bidirektionalem Datenaustausch vorgenommen werden muss.

### 3. Modell für Multiscreen-Interaktionen

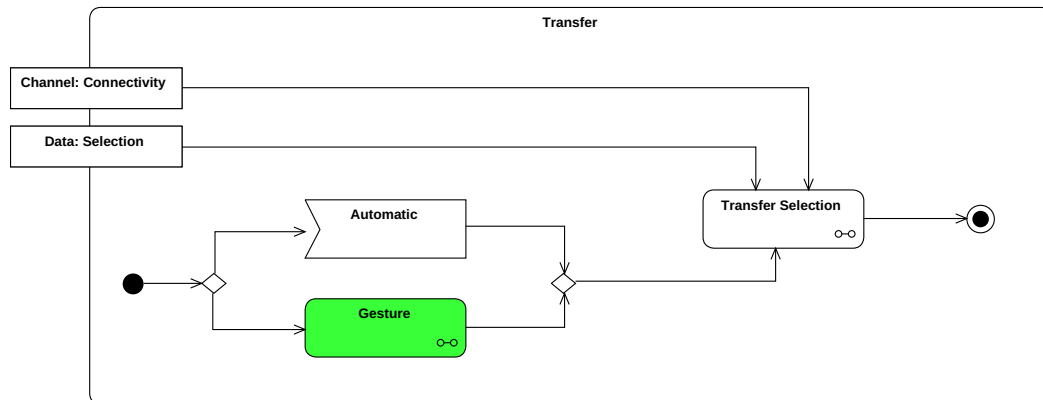


Abbildung 3.21.: Beginn des Transfers mit Vorbedingungen und Auslösern.

- die Übertragung wird automatisch gestartet, wenn keine explizite Geste mehr erforderlich ist (z. B. dann, wenn nur ein *Command* übertragen wird).

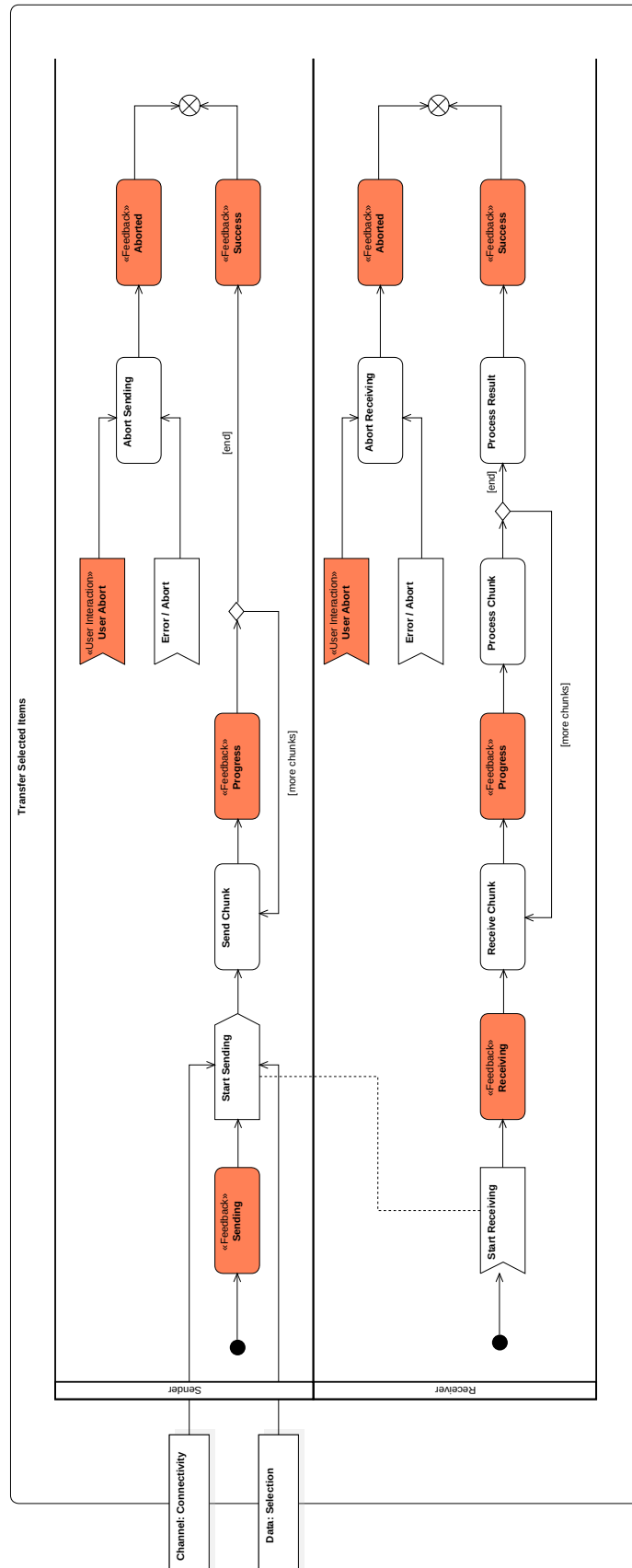
Sobald beide Voraussetzungen erfüllt sind und einer der Auslöser erkannt wurde, wird die Datenübertragung in der Aktivität *Transfer Selected Items* durchgeführt. Die Details dieses Ablaufes werden durch das Modell in Abbildung 3.22 näher beschrieben.

Das sendende Gerät (*Sender*) informiert den Benutzer über den Start der Datenübertragung und fängt an, Daten an den Empfänger (*Receiver*) zu senden. Dieser informiert ebenfalls mit einem System-Feedback über den Start des Empfangs und verarbeitet die eintreffenden Datenpakete (*Chunks*) kontinuierlich. Für jeden erhaltenen *Chunk* gibt es eine Rückmeldung über den Fortschritt (*Feedback Progress*) sowie einen eventuellen Verarbeitungsschritt (*Process Chunk*), wenn z. B. ein Datenstrom empfangen wird. Ebenso gibt es auf der Senderseite kontinuierliches Feedback über den Fortschritt.

Sowohl Sender als auch Empfänger ermöglichen einen Abbruch der Übertragung durch den Nutzer (Event *User Abort*) bzw. können einen Fehler oder Abbruch der Übertragung feststellen (Event *Error / Abort*). In beiden Fällen kann ein Feedback über den Abbruch des Transfers stattfinden. Wird die Übertragung hingegen erfolgreich beendet, findet auf dem Empfänger eine Verarbeitung des Endergebnisses statt, in der z. B. eine empfangene Datei angezeigt oder ein Befehl ausgeführt wird. Auf der Senderseite ist dagegen nur ein Feedback über den Erfolg des Transfers vorgesehen, da keine zu verarbeitenden Daten anfallen.

Ein Beispiel für die Gestaltung dieses Ablaufes findet sich bei [Hinckley, 2003], in dem eine geöffnete Webseite geteilt werden soll. Durch eine Bump-Geste soll die





**Abbildung 3.22.:** Ablauf eines Datentransfers in der Aktivität Transfer Selected Items.

auf einem Gerät geöffnete Webseite auch auf dem anderen Gerät geöffnet werden. Hier sind alle Voraussetzungen zum Start des Transfers erfüllt:

- Eine (implizite) *Selection* in Form der geöffneten Webseite,
- eine Verbindung zwischen beiden Geräten (*Channel*), die durch den Bump zustande kommt (siehe Kapitel 3.3.4) und
- die Bump-Geste als Auslöser für die Übertragung.

Hier kommt die Variante aus Abbildung 3.5 zum Einsatz, in der die Verbindung und der Transfer durch die gleiche Geste erfolgen. Somit löst die Bump-Geste den Verbindungsaufbau als auch den Transfer aus. Der vorgeschlagene Ablauf des Transfers lässt sich durch das Modell erfassen:

- Vor dem Transfer wird ein „Teleportier“-Sound abgespielt [Hinckley, 2003], zudem werden Pfeile in Richtung der Bildschirmränder angezeigt (*Feedback Starting / Feedback Receiving*).
- Ein *Command* wird übertragen, der die URL der geöffneten Webseite beinhaltet (*Send Chunk / Receive Chunk*). In diesem Fall wird die Aktivität *Process Chunk* nicht ausgeführt, da der Befehl erst ausgeführt werden kann, wenn er vollständig übertragen wurde.
- Der *Receiver* führt den Befehl aus, indem der Browser geöffnet und die Webseite angezeigt wird (*Process Result*).

Da das Übertragen eines *Commands* aufgrund des geringen Datenvolumens relativ schnell geht, ist in [Hinckley, 2003] kein Feedback für Übertragungsprobleme oder Fehlerfälle vorgesehen, es gibt auch kein Feedback über den Fortschritt. Sofern eine Verbindung besteht und die Übertragung gestartet wird, geht Hinckley vermutlich davon aus, dass diese auch erfolgreich beendet werden kann.

Ebenso wird in [Hinckley, 2003] eine Variante des Systems vorgestellt, in der ein Whiteboard von beiden Nutzern geteilt und gleichzeitig auf beiden Geräten verwendet wird. Auch diese Variante lässt sich mittels des Modells beschreiben:

- Die *Selection* ist in diesem Fall vom Typ *Stream*.
- Die Aktivität *Process Chunk* verarbeitet den kontinuierlich eintreffenden Datenstrom.

- Die Kollaboration wird beendet, wenn ein Benutzer die Verbindung explizit abbricht (*User Abort*) oder den Raum verlässt, wodurch das Event *Error / Abort* ausgelöst wird.
- Die Aktivität *Process Result* wird nicht benötigt, da nach dem Abbruch der Kollaboration keine abschließende Verarbeitung der Stream-Daten notwendig ist.

Auch die Datenübertragungen des Systems von Dachzelt und Buchholz (s. auch Kapitel 3.3.1) können durch das Modell beschrieben werden [Dachzelt und Buchholz, 2009]. Zunächst sollen Bilder durch eine Wurf-Geste vom Smartphone auf das Smart TV übertragen werden. Die drei Bedingungen des Modells in Abbildung 3.21 werden dazu wie folgt definiert:

- Die *Selection* ist vom Typ *MultipleFiles* und wird implizit erstellt, wenn der Benutzer Fotos in seiner Applikation geöffnet hat.
- Der *Channel* ist in Form einer Bluetooth- oder Wi-Fi-Verbindung gegeben, die bereits beim Betreten des Raumes aufgebaut wird.
- Der Auslöser für den Transfer ist eine Wurf-Geste.

Für die Übertragung von Daten vom Smart TV auf das Smartphone würde ein ähnlicher Ablauf modelliert. Lediglich Sender und Empfänger wären vertauscht und die Wurf-Geste durch eine Fetch-Back-Geste ersetzt (siehe Kapitel 3.3.1).

Wird ein Transfer per Wurf-Geste gestartet, werden alle Bilder auf das Smart TV (*Receiver*) übertragen. Dachzelt und Buchholz geben hier keine Gestaltungshinweise für den Ablauf der Übertragung, denkbar wäre aber z. B. eine Fortschrittsanzeige während des Sendens, da die Menge der gesendeten Bilder bekannt ist. Allerdings finden sich Gestaltungshinweise für den Abschluss des Transfers, den die Anzeige der Bilder als Galerie in der Aktivität *Process Result* bildet. Auf dem sendenden Smartphone wäre ebenfalls ein *Feedback Success* denkbar.

Zum Durchschalten der Bilder wiederholt sich die *Transfer*-Aktivität, allerdings diesmal in einer anderen Ausprägung:

- Der *Channel* ist weiterhin verfügbar.
- Die *Selection* ist vom Typ *Command* („Weiterschalten zum nächsten Bild“).
- Der Auslöser ist *Automatic*, da die Auswahl des Befehls bereits die Geste enthält.

Die anschließende Übertragung des Befehls ist ähnlich dem der geteilten Webseite per Bump-Geste. Hier ist weniger Feedback nötig, in der Aktivität *Process Result* wird lediglich das nächste Foto in der Galerie angezeigt.

Es werden so lange *Select*- und *Transfer*-Aktivitäten durchgeführt, bis die Verbindung zwischen Geräten getrennt wird. Sollen keine weiteren Transfers stattfinden, kann die Verbindung entweder aktiv durch den Benutzer oder implizit getrennt werden. Die Modellierung dazu findet sich im folgenden Kapitel.

#### 3.3.7. Disconnect

Die letzte Aktivität im Lebenszyklus einer Multiscreen-Anwendung ist der *Disconnect*, das Trennen der Verbindung zwischen zwei Geräten (s. Rahmenmodell in Kapitel 3.3.1). Im Vergleich zum Verbindungsaufbau, der verschiedene Fehlerfälle, Feedbacks und Nutzerinteraktionen vorsieht, läuft der Verbindungsabbau relativ einfach ab. In Abbildung 3.23 ist der Ablauf beschrieben.

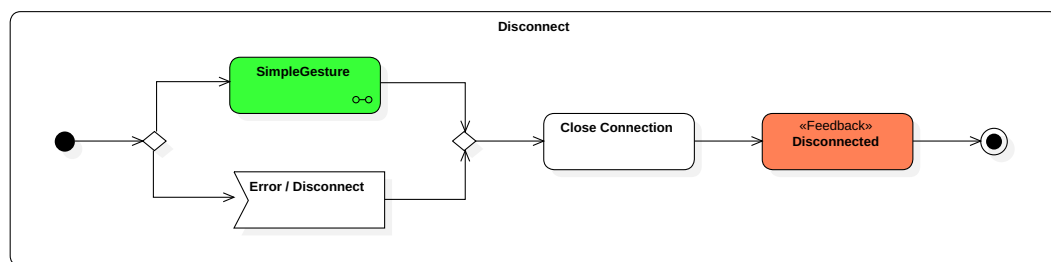


Abbildung 3.23.: Trennen der Verbindung in der Aktivität *Disconnect*.

Angestoßen wird der Verbindungsabbau je Gerät entweder durch eine einfache Geste oder durch ein Systemevent, das über einen Verlust der Verbindung informiert (z. B. beim Verbindungsabbau durch das verbundene Gerät oder einen Verbindungsfehler). Anschließend wird die Verbindung geschlossen und der Benutzer darüber informiert (*Feedback Disconnect*). Eine synchrone Geste ist an dieser Stelle nicht sinnvoll, da das Schließen der Verbindung sonst nicht von einem Nutzer alleine durchgeführt werden könnte. Würde der Nutzer der Gegenseite seinen Teil der synchronen Geste nicht erfüllen, könnte er das Trennen der Verbindung verhindern.

In der Literatur finden sich einige Hinweise für die (gestenbasierte) Gestaltung der *Disconnect*-Aktivität, wobei das Spektrum der verwendeten Gesten deutlich eingeschränkter als bei der *Connect*-Aktivität ist.

Um zwei Geräte zu trennen, die per Bump verbunden wurden, schlägt Hinckley ein Auseinanderbewegen der Tablets vor [Hinckley, 2003]. Sobald ein Tablet erkennt, dass es von dem verbundenen Tablet wegbewegt wird, informiert es dieses über den Verbindungsabbau und gibt ein visuelles (in Form eines zerbrochenen Pfeils, entsprechend dem visuellen Feedback beim *Connect*, siehe Kapitel 3.3.4) sowie akustisches Feedback in Form eines „breaking sounds“. Beim Erhalt der Nachricht zum Disconnect wiederholt das zweite Tablet beide Feedbacks und baut ebenfalls die Verbindung ab.

Für die Stitch-Geste schlagen Hinckley und Kollegen vor, den Verbindungsabbau einfach über die Auswahl eines Menüpunktes in der Nutzeroberfläche vorzunehmen und verzichten gänzlich auf den Einsatz einer umfangreichen Geste [Hinckley u. a., 2004]. Hier würde entsprechend dem Modell in Abbildung 3.23 einfach auf ein Systemevent reagiert, das zweite Gerät würde durch einen Fehler bzw. Timeout über den Verbindungsabbau informiert.

In [Dachselt und Buchholz, 2009] wird kein Mechanismus zum Verbindungsabbau explizit vorgeschlagen. Allerdings wird die Verbindung per Bluetooth oder Wi-Fi beim Betreten des Raumes aufgebaut, was nahelegt, dass das Verlassen des Raumes entweder als Geste zum Verbindungsabbau interpretiert werden kann oder ein Verlust des Funksignals implizit die Verbindung trennt. Beides ist durch das Modell in Abbildung 3.23 abbildbar.

Lucero und Kollegen schlagen vor, ein Smartphone durch Aufnehmen von einem Tisch von verbundenen Geräten zu trennen oder das Teilen von Fotos durch Kippen des Smartphones (Tilt-Geste) zu beenden [Lucero u. a., 2011]. Im ersten Fall wird ein visuelles Feedback angezeigt, für den zweiten Fall schlagen sie ein langes Vibrieren vor. Nimmt der Nutzer sein Gerät vom Tisch, auf dem eine Multiscreen-Interaktion stattgefunden hat, würde man hier eine auf Distanz basierende Geste definieren. Zwar sind die Geräte noch in Sendereichweite, allerdings kann z. B. per NFC oder BLE erkannt werden, wann der Abstand einen Schwellwert erreicht, ab dem der Verbindungsabbau ausgelöst wird. Die Tilt-Geste kann ebenfalls als einfache Geste definiert und hier als Auslöser für den Verbindungsabbau eingesetzt werden.

Zusammenfassend lässt sich sagen, dass zwar teilweise einfache Gesten zum Einsatz kommen, um die *Disconnect*-Aktivität zu starten, in vielen Fällen allerdings ein Verlassen des Sendebereiches implizit für das Trennen der Verbindung sorgt.

Soll eine festgelegte Distanz zwischen Geräten den Verbindungsabbau auslösen, muss das entsprechend dem „Leave To Disconnect“ Pattern aus der SysPlace-Patternsammlung (siehe Anhang B) explizit modelliert werden, um z. B. mittels *Constraints* auf entsprechende Distanzen reagieren zu können.

#### 3.3.8. Statische Sicht – Domänenmodell

Abschließend wird in diesem Kapitel eine statische Sicht des Modells vorgestellt. Während in den vorigen Kapiteln die dynamischen Aspekte von Multiscreen-Anwendungen durch das Modell erfasst wurden, wird nun als Ergänzung ein Domänenmodell vorgestellt, das die in den Ablaufdiagrammen verwendeten Komponenten erfasst und in Beziehung zueinander setzt.

Abbildung 3.24 zeigt die statische Modellierung von Gesten und den zur Erkennung von Gesten notwendigen Komponenten. Gesten tauchen in jeder Aktivität des Rahmenmodells auf, zum Verbinden / Trennen von Geräten, Auswählen von Dateien oder zum Starten von Dateitransfers. Kapitel 3.3.2 und Kapitel 3.3.3 beschreiben die Gestenerkennung im Detail und führen Begriffe wie *GestureEvent* und *Constraint* ein, die in der statischen Sicht wiederzufinden sind. Im Zentrum steht dabei jeweils der allgemeine Begriff der Geste, der weiter in einfache (*SimpleGesture*) und synchrone (*SynchronousGesture*) Gesten unterteilt wird, für die es jeweils konkrete Ausprägungen analog zum SysPlace-Patternkatalog bzw. der analysierten Literatur gibt. Es besteht zudem eine Abhängigkeit zwischen *SynchronousGesture* und *SimpleGesture*. Da die Erkennung einer synchronen Geste immer die Erkennung von zwei einfachen Gesten voraussetzt, wird diese als Komposition einfacher Gesten abgebildet.

Im rechten Teil des Diagramms finden sich zudem die Konzepte von *Constraints* und *GestureEvents* wieder. Wird eine Geste erkannt, wird ein sog. *GestureEvent* erstellt, das spezifische Parameter über die Geste bereitstellt (wie z. B. Anfangs- und Endkoordinaten einer Swipe-Geste). Diese Parameter können mit vordefinierten *Constraints* verglichen werden, um eine Geste mit Qualitätskriterien auszustatten, die zum jeweiligen Anwendungsfall passen.

Zudem sind zur Erkennung einer konkreten Geste immer bestimmte Geräteeigenschaften wie vorhandene Sensoren notwendig, die hier als *Capabilities* erfasst wurden und den *required*-Annotationen der Gestenerkennung entsprechen (s. Modell in Abbildung 3.7). Dadurch lassen sich durch das statische Modell Abhängigkeiten

zwischen verwendeten Erkennungsalgorithmen und benötigter Hard- oder Software herstellen, wodurch eine der Kernanforderungen an das Modell erfüllt wird.

Das Spektrum möglicher *Capabilities* wird in Abbildung 3.25 modelliert. Der allgemeine Begriff der *Capability* aus Abbildung 3.24 wird hier in Kategorien unterteilt:

- *Input*: Erfassen von direkten oder indirekten Nutzereingaben.
- *Output*: Ausgabe von Systemreaktionen.
- *Proximity*: Wahrnehmen von Geräten in der Umgebung.
- *Connectivity*: Herstellen von Netzwerkverbindungen.

Als *Input* gelten dabei alle Geräteeigenschaften, die in irgendeiner Form Eingaben des Nutzers verarbeiten. Während Tastatur oder Maus direkte Nutzereingaben ermöglichen, erfassen z. B. Beschleunigungssensoren auch Eingaben, die der Nutzer eventuell unbewusst tätigt. Die *Capabilities* des Modells zur Gestenerkennung in Abbildung 3.24 sind zumeist vom Typ *Input*, oft in Form eines Touchscreens (z. B. zur Swipe-Erkennung) oder eines Bewegungs- und Lagesensors.

Unter *Output* werden alle Geräteeigenschaften zusammengefasst, die Ausgaben des Systems ermöglichen (s. auch Kapitel 2.4.1), wie z. B. Bildschirm, Vibration oder Sound. Diese sind vor allem für die Definition der *Feedback*-Aktivitäten innerhalb der Ablaufbeschreibungen relevant.

Eigenschaften vom Typ *Connectivity* spezifizieren den *Channel*, der in den Aktivitäten *Connect*, *Transfer* und *Disconnect* des Rahmenmodells eine wichtige Rolle für Multiscreen-Interaktionen spielt. Die Art des *Channels* legt fest, welche Infrastruktur für die Realisierung von Multiscreen-Anwendungen benötigt wird. Entsprechend Kapitel 2.4.1 wird zwischen Ad-Hoc- und Infrastruktur-Netzwerkfähigkeit unterschieden.

Die Menge aller Geräteeigenschaften, die von einer Multiscreen-Anwendung benötigt werden, ergeben als Komposition ein *Device*, das die notwendige Hard- und Software für die Anwendung bereitstellt. Ein Gerät kann demzufolge allgemein als eine Menge benötigter Eigenschaften modelliert werden.

Abbildung 3.26 zeigt den Zusammenhang zwischen *User Interactions*, *Feedback* und *Capabilities*. Die verschiedenen farblich markierten *User Interaction*- und *Feedback*-Aktivitäten innerhalb der Ablaufmodelle lassen sich so analog zu den Gesten mit benötigten Geräteeigenschaften in Zusammenhang bringen. Da Systemfeedback und Nutzereingaben auch multimodal sein können, wird eine *User Interac-*

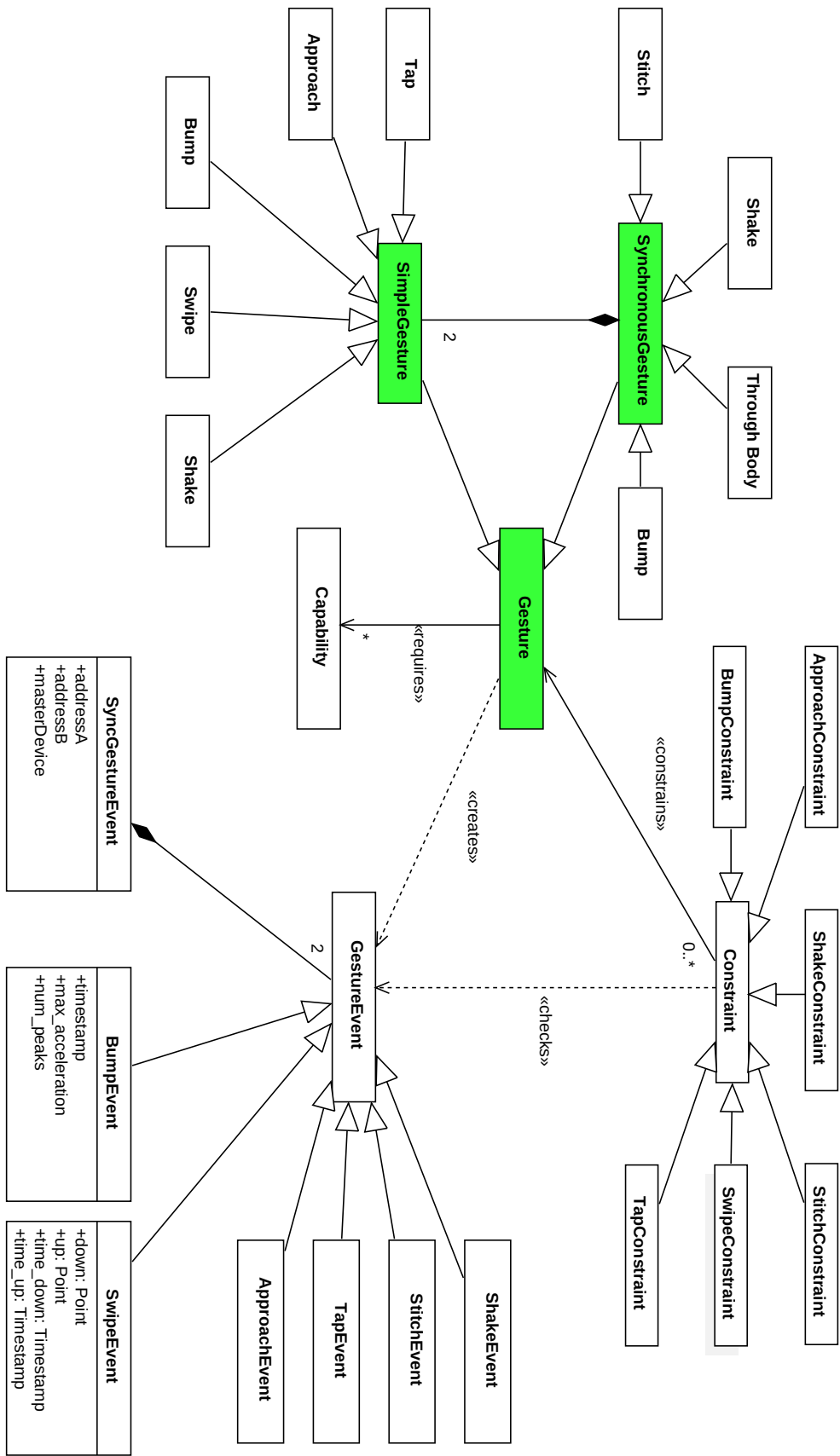


Abbildung 3.24.: Domänenmodell für die Gestenerkennung.



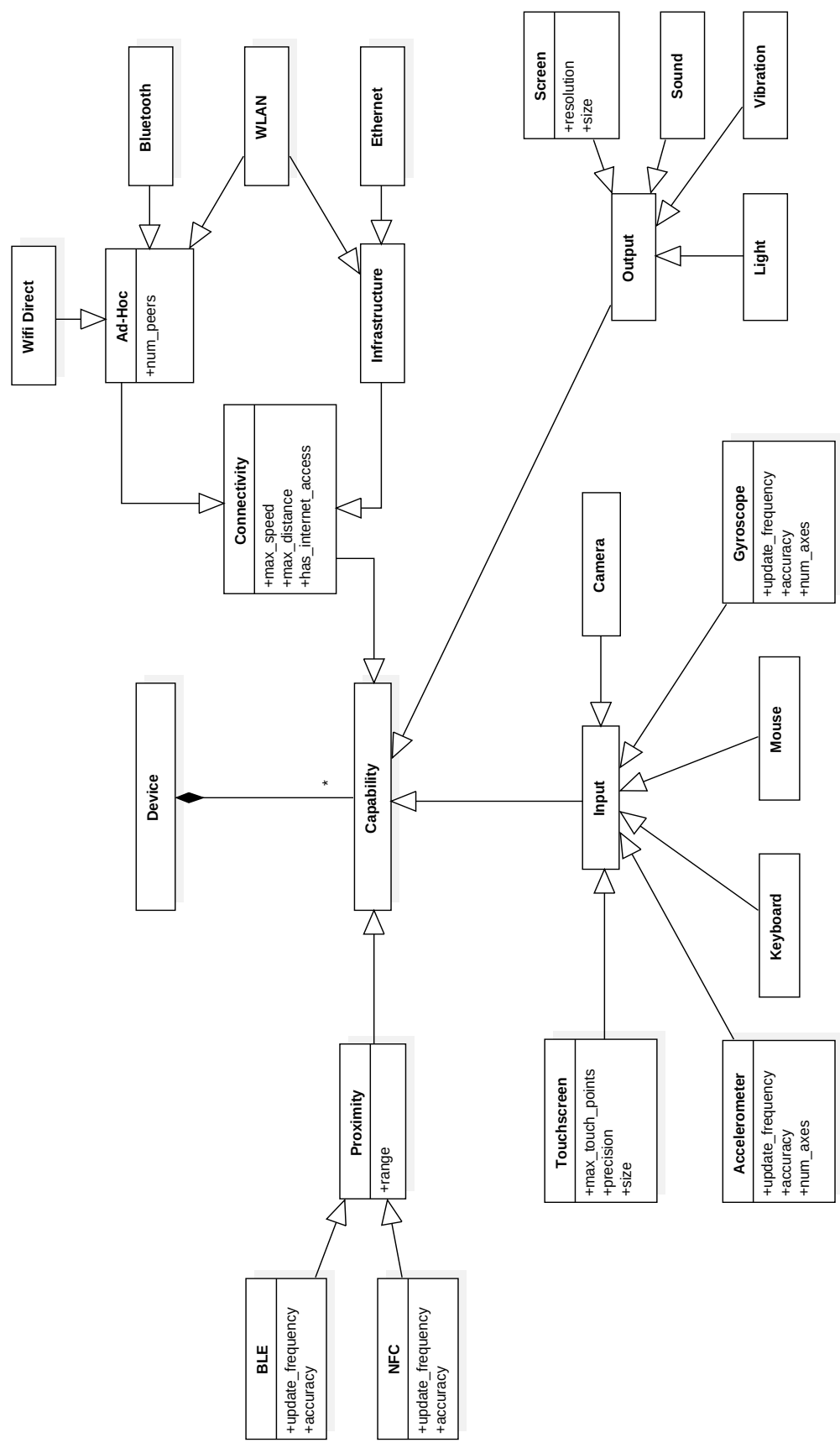
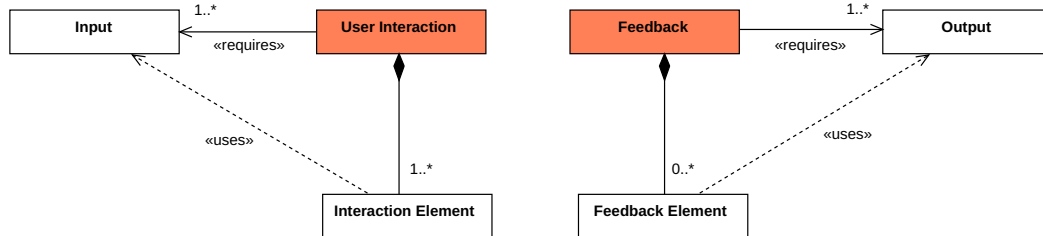


Abbildung 3.25.: Domänenmodell der Geräteeigenschaften (Capabilities).

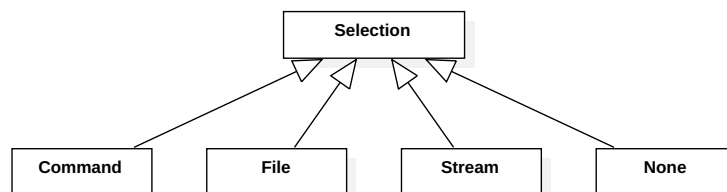
### 3. Modell für Multiscreen-Interaktionen

tion bzw. ein *Feedback* hier als Komposition einzelner Elemente beschrieben, die jeweils eine *Capability* des Gerätes voraussetzen. Die Gesamtheit aller *User Interactions* und *Feedbacks* in einer Multiscreen-Anwendung hat somit wiederum einen Einfluss auf das *Device* in Abbildung 3.25.



**Abbildung 3.26.:** Aufbau von *User Interactions* und *Feedback* mit Bezug zu *Capabilities*.

Der letzte Aspekt, der durch das statische Modell der Vollständigkeit halber noch erfasst werden muss, ist die *Selection*, die in den Aktivitäten *Select* und *Transfer* des Rahmenmodells eine Rolle spielt und nach Kapitel 3.3.5 die Ausprägungen *Command*, *File*, *MultipleFiles* oder *Stream* haben kann, wobei *MultipleFiles* eine Aggregation vom Typ *File* ist und hier nicht separat aufgeführt wird. Abbildung 3.27 zeigt das entsprechende Domänenmodell dazu.



**Abbildung 3.27.:** Domänenmodell für die *Selection*.

Diese statischen Sichten sind eine wichtige Ergänzung des Modells, da sie es um einige Möglichkeiten erweitern. Sie verdeutlichen die Wiederverwendbarkeit von Komponenten, die nur einmal modelliert werden müssen, um sie in die verschiedenen dynamischen Sichten einfügen zu können. Zudem sind sie ein Ansatzpunkt zur Erweiterung des Modells, indem z. B. neue Technologien zur Ein- und Ausgabe oder neue Gesten in die statische Struktur aufgenommen und in den dynamischen Modellen verwendet werden können.

Es ist außerdem möglich, aus einem dynamischen Modell das zugehörige statische Modell abzuleiten oder umgekehrt. So kann eine Multiscreen-Anwendung beliebig dynamisch modelliert und anschließend jede Komponente statisch erfasst werden, um Rückschlüsse über Mindestanforderungen an Geräte zu ziehen oder benötigte

Parameter zusammenzufassen. Umgekehrt ist es möglich, das statische Modell als Basis vorab zu definieren und die dynamischen Aspekte daraus abzuleiten. Es ist z. B. denkbar, dass nur ein bestimmter Satz an Geräteeigenschaften zur Verfügung steht und die Design-Möglichkeiten der dynamischen Aspekte einer Multiscreen-Anwendung einschränkt.



## **Kapitel 4**

# **Validierung des Modells**

In diesem Kapitel wird eine Bewertung des erarbeiteten Modells aus Kapitel 3 vorgenommen. Dazu wird zunächst eine qualitative Aussage getroffen, zu welchem Grad die Anforderungen, die in Kapitel 3.2 formuliert wurden, durch das Modell erfüllt werden. Anschließend werden zwei Gruppen von Patterns aus dem SysPlace-Patternkatalog, basierend auf der Bump- bzw. Approach-Geste, exemplarisch durch das Modell beschrieben. Die Flexibilität des Modells soll dadurch verdeutlicht werden, dass beide Gruppen unterschiedliche Technologien voraussetzen und unterschiedliche Einsatzgebiete haben und dennoch einheitlich erfasst werden können. Darauf aufbauend wird diskutiert, inwiefern sich von diesen Beispielen auf die Anwendungsmöglichkeiten für die übrigen Patterns des SysPlace-Patternkataloges schließen lässt.

### **4.1. Qualitative Bewertung der Anforderungserfüllung**

Die Anforderungen an das Modell aus Kapitel 3.2 lassen sich in drei Gruppen zusammenfassen. Es gibt Anforderungen an

- die Erfassung der Gestenerkennung,
- die Beschreibung der anschließenden Systemreaktion und
- die Verwendbarkeit des Modells.

In diesem Kapitel wird gezeigt, welche Aspekte des Modells zur Erfüllung einzelner Anforderungen dieser Gruppen beitragen.

#### 4. Validierung des Modells

---

Für die Gestenerkennung werden in den Anforderungen einige Aspekte definiert, die durch das Modell abbildbar sein sollen:

- Direkte oder indirekte Eingaben des Nutzers,
- abstrakte Beschreibung des Algorithmus und mögliche Alternativen,
- Einschränkungen der Gestenerkennung,
- Hard- und Softwarevoraussetzungen für die Gestenerkennung,
- Kommunikationsprotokolle für synchrone Gesten und
- Wiederverwendbarkeit von Teilkomponenten.

In Kapitel 3.3.2 wird das Modell für die Beschreibung einfacher Gesten vorgestellt, welches aus zwei Teilen besteht (siehe Abbildung 3.6):

1. *Recognize Gesture*: Eine Geste aufgrund von Sensordaten erkennen.
2. *Check Constraints*: Einschränkungen der erkannten Geste prüfen.

Die Aktivität *Recognize Gesture* beschreibt abstrakt den Erkennungsalgorithmus für eine Geste (siehe Abbildung 3.7 für eine exemplarische Anwendung) mittels eines Zustandsdiagramms. Hard- und Software-Voraussetzungen werden in Form von Parametern und benötigten Sensoren als Annotationen beschrieben. Die Eingaben des Nutzers werden als System-Events dargestellt, die Transitionen zwischen den Zuständen auslösen. Berechnungen, die bei diesen Transitionen ausgeführt werden, können über Pseudo-Code-Annotationen abstrakt beschrieben werden.

Das Ergebnis einer Gestenerkennung ist ein *GestureEvent*, das Informationen über die erkannte Geste beinhaltet, die im nächsten Schritt (*Check Constraints*) geprüft werden können. Dadurch wird die Anforderung an die Einschränkung erkannter Gesten erfüllt, zudem wird durch die Trennung zwischen Erkennung und Einschränkung eine Wiederverwendbarkeit der einzelnen Teile erreicht. Der Erkennungsalgorithmus kann beliebig ausgetauscht werden, sofern das gleiche *GestureEvent* an die Aktivität *Check Constraints* übergeben wird.

Entsprechend den Anforderungen wird in Kapitel 3.3.3 ein Kommunikationsprotokoll beschrieben, das der einfachen Gestenerkennung noch einen Vergleich zweier *GestureEvents* hinzufügt (*Gesture Event Matching*) und dadurch die Erkennung synchroner Gesten ermöglicht. Es wurde ein genereller Ablauf für alle synchronen Gesten erarbeitet, der eine hohe Austauschbarkeit der einfachen Gestenerkennung und der Vergleichsaktivität ermöglicht. Sofern die zwei *GestureEvents*, also die

Outputs der einfachen Gestenerkennungen, gleich den zwei Inputs des Vergleichsschritts sind, kann eine beliebige Kombination einfacher zu synchronen Gesten vorgenommen werden, wodurch wiederum eine hohe Wiederverwendbarkeit von Teilkomponenten für die synchrone Gestenerkennung gegeben ist.

Hinsichtlich der verschiedenen Systemreaktionen, die auf das Erkennen einer einfachen oder synchronen Geste folgen, wurden in den Anforderungen folgende Aspekte genannt, die durch das Modell darstellbar sein müssen:

- Vollständige Ablaufbeschreibung,
- Vor- und Nachbedingungen bzw. Objektfluss,
- Interaktionsmöglichkeiten des Nutzers,
- Rückmeldungen (Feedback) des Systems,
- Implikation der gewählten Geste auf den Ablauf und
- Alternativen und Fehlerfälle.

In Kapitel 3.3.1 wurde zunächst ein Überblick über den Lebenszyklus einer Multi-screen-Anwendung gegeben, die Kapitel zu den Aktivitäten *Connect* (3.3.4), *Select* (3.3.5), *Transfer* (3.3.6) und *Disconnect* (3.3.7) vertiefen diese Sicht jeweils um detaillierte Beschreibungen in Form von Ablaufdiagrammen. Jede dieser Aktivitäten kann mit Vorbedingungen (z. B. erforderlicher Auslöser oder benötigter Input) und Nachbedingungen (z. B. ausgewählte Daten oder geöffnete Verbindungen) versehen werden. Platzhalter innerhalb der Ablaufbeschreibungen beschreiben die Stellen, an denen der Nutzer mit dem System interagieren kann (*User Interaction*), bzw. an der das System dem Nutzer eine Rückmeldung (*Feedback*) geben kann. Sofern die Wahl zwischen einfacher oder synchroner Geste eine Implikation auf den Ablauf hat, wurden beide Alternativen modelliert. Mögliche Fehlerfälle wurden in Form von System-Events erfasst, deren Behandlung ebenfalls Teil der Ablaufbeschreibung ist und Feedback beinhalten kann.

An die Verwendbarkeit des Modells gibt es zusammengefasst folgende Anforderungen:

- Wahl einer verständlichen Repräsentation für Entwickler und Interaktionsdesigner,
- Verfügbarkeit von Tools zur Arbeit mit dem Modell,
- Aussagefähigkeit über Mindestanforderungen an Hard- und Software,

- einfache Erweiterbarkeit und
- Einsatz als Kommunikationsmittel.

Durch die Wahl der UML als Modellierungssprache werden die ersten beiden Anforderungen adressiert, wie schon in der Einleitung zu Kapitel 3 dargelegt wurde.

Um eine Aussage über die Mindestanforderungen an Hard- und Software treffen zu können, wurden die dynamischen Modelle in Kapitel 3.3.8 um eine statische Sicht, das Domänenmodell, erweitert. Indem man alle verwendeten Entitäten der erstellten Ablaufbeschreibungen in einer statischen Sicht aggregiert, kann abstrakt die benötigte Umgebung für den Einsatz des beschriebenen Patterns abgeleitet werden. Umgekehrt ist es auch möglich, die verfügbare Infrastruktur zunächst im Domänenmodell zu erfassen, um die Gestaltungsmöglichkeiten der dynamischen Modelle auf diesen Ausschnitt zu beschränken.

Auch die Erweiterbarkeit des Modells wird über die statische Sicht realisiert. Werden z. B. eine neue Übertragungstechnologie oder ein weiteres Ausgabegerät verfügbar, können diese in das Domänenmodell aufgenommen und an den entsprechenden Stellen im dynamischen Modell eingesetzt werden. So können z. B. neue Sensoren alternative Möglichkeiten zur Gestenerkennung eröffnen, die wiederum dynamisch erfasst und als Alternative zur bisherigen Gestenerkennung dokumentiert werden können.

Erstellte Modelle können als Dokumentation und somit als Kommunikationsmittel für die Gestaltung von Multiscreen-Interaktionen verwendet werden. Indem die wichtigen Gestaltungsentscheidungen wie Erkennungsalgorithmen oder Feedbacks als austauschbare Elemente in einen übergeordneten Ablauf eingesetzt werden, können mittels des Modells Fragestellungen wie die folgenden diskutiert werden:

- Wie wäre es, wenn wir als Erfolgs-Feedback an *dieser Stelle* noch zusätzlich den Sound abspielen, den wir auch *hier* verwenden?
- Können wir *diese Geste* auch einsetzen, wenn kein Beschleunigungssensor verfügbar ist?
- Würde die Übertragung beschleunigt, wenn wir *den Übertragungskanal* von Bluetooth auf Wi-Fi Direct umstellen?
- An welchen Stellen wird in der Anwendung *diese Dialogbox* angezeigt?

Neben den visuellen Aspekten der Diagramme sind dabei auch die eingeführten Begrifflichkeiten des Domänenmodells hilfreich. Die Verwendung bzw. individu-



elle Anpassung der verwendeten Begrifflichkeiten fördert eine einheitlich genutzte und verstandene Sprache im Projektteam. Um die technische Realisierung erstellter Modellinstanzen zu unterstützen, wird in Kapitel 4.4 ein Ansatz zur Entwicklung eines Frameworks vorgestellt, das die Übersetzung modellierter Konzepte in ausführbaren Code erleichtern könnte.

## 4.2. Exemplarische Instanziierungen des Modells

### 4.2.1. Bump-Patterns

In diesem Kapitel wird das Modell zur technischen Beschreibung aller Patterns angewendet, die auf einer Bump-Geste basieren (siehe Anhang B):

- Bump To Connect
- Bump To Give
- Bump To Take
- Bump To Exchange

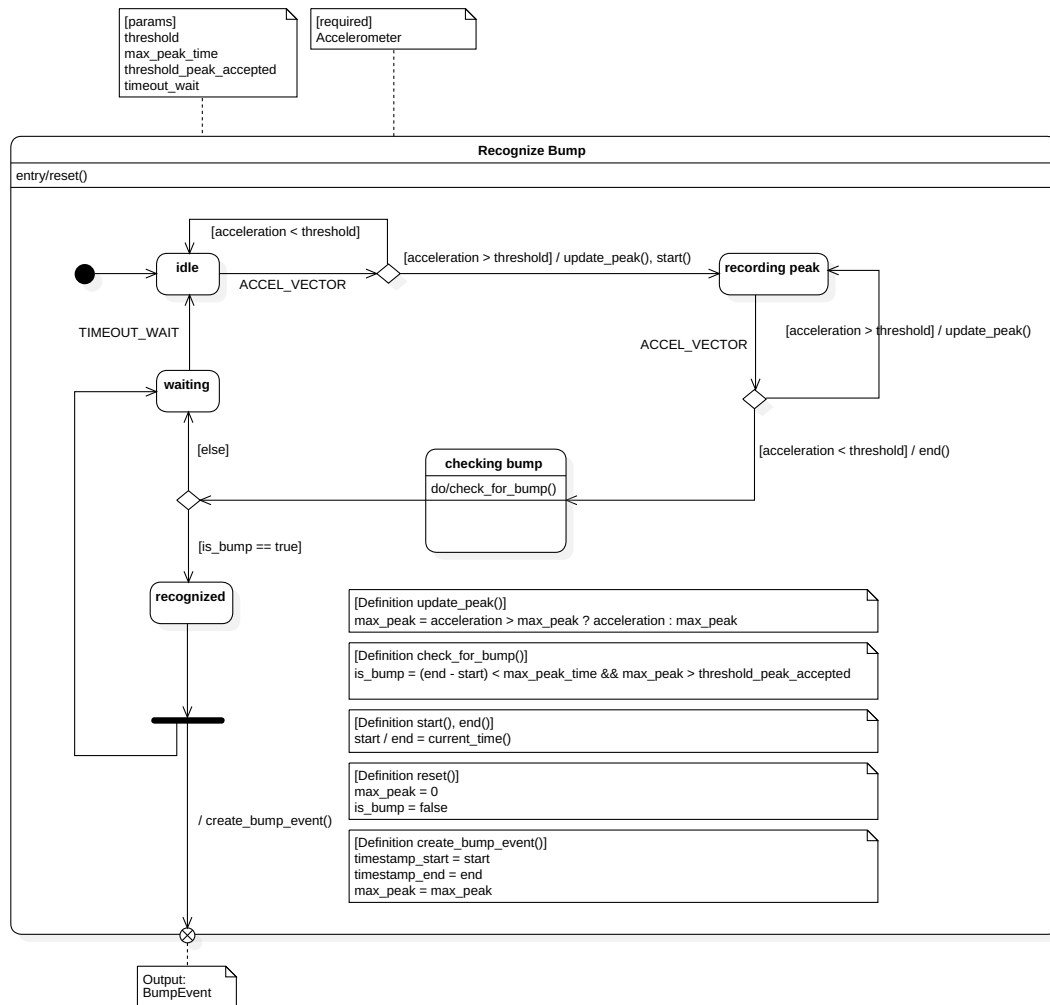
Die Ausgestaltung der Details ist dabei stark angelehnt an die Ideen in [Hinckley, 2003], wobei auch Variationen der Details denkbar sind. Es wird lediglich versucht, eine exemplarische, konsistente Beschreibung einer Gruppe von Patterns zu erstellen.

Grundlegend für alle vier Patterns ist die Beschreibung der verwendeten Bump-Geste. Dem Modell in Abbildung 3.6 entsprechend muss dazu zunächst die Erkennung der Geste auf einem Gerät beschrieben werden. Abbildung 4.1 zeigt eine detaillierte Beschreibung der Bump-Geste mittels des Modells.

Der Algorithmus basiert auf der Erkennung eines sog. *Peaks* (deut. Spitze) in den Beschleunigungswerten. Vorausgesetzt werden dafür ein Beschleunigungssensor und vier Parameter:

- *threshold*: Schwellwert, ab dem vom Auftreten eines Peaks ausgegangen wird.
- *max\_peak\_time*: Maximale Zeit zwischen Übertretungen des Schwellwerts, also die „Breite“ des Peaks.

## 4. Validierung des Modells



**Abbildung 4.1.:** Erkennung einer Bump-Geste entsprechend der *Gesture Detection* in Abbildung 3.6.

- *threshold\_peak\_accepted*: Schwellwert, ab dem ein Peak akzeptiert wird (Beschleunigungswert, der innerhalb des Peaks mindestens überschritten werden muss).
- *timeout\_wait*: Wartezeit nach Bump-Erkennung.

Im Anfangszustand werden so lange Beschleunigungsvektoren gelesen, bis der *threshold*-Wert überschritten wurde. Anschließend wird bis zum nächsten Überschreiten der Schwelle gewartet, also dem Ende des Peaks. Dazwischen wird die Maximalbeschleunigung überschrieben (in *update\_peak()*), wenn ein höherer Wert als der bisher maximale gelesen wurde. Nach erneutem Unterschreiten der Schwelle wird geprüft, ob die Spitze das zeitliche Fenster *max\_peak\_time* nicht über- und den erforderlichen Schwellwert für die Beschleunigung nicht unterschritten hat. Sind

diese Bedingungen erfüllt, wird die Bump-Geste erkannt und das entsprechende *BumpEvent* generiert, ansonsten beginnt die Erkennung von vorne.

In beiden Fällen wird eine Wartezeit von *timeout\_wait* abgewartet, bevor die Erkennung erneut beginnt, um zu schnelle Folgen von Bump-Erkennungen auszuschließen. Für die Bump-Erkennung wurden keine Constraints verwendet, dies wäre allerdings z. B. für die minimal zulässige Beschleunigung denkbar, je nachdem, ob dieser Wert individuell einstellbar oder fest sein soll.

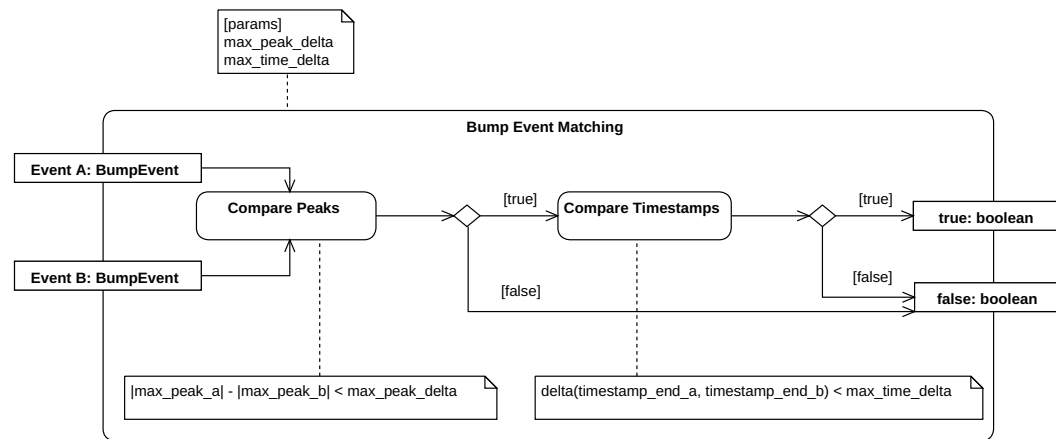


Abbildung 4.2.: Erkennung einer synchronen Bump-Geste entsprechend dem Modell in Abbildung 3.10.

Da es sich beim Bump um eine synchrone Geste handelt, muss noch der Vergleich zweier Bump-Events entsprechend dem Modell in Abbildung 3.10 beschrieben werden. Abbildung 4.2 zeigt den Vergleich von zwei Bump-Events anhand der maximalen Beschleunigung an der Spitze (mit einer erlaubten Differenz von *max\_peak\_delta*) und dem zeitlichen Abstand beider Bump-Events (mit einer maximalen Differenz von *max\_time\_delta*). Sind beide Bedingungen erfüllt, wird die synchrone Geste erkannt und beide Geräte über ihren Kommunikationspartner informiert.

Solch eine Erkennung ist auch in den alternativen Ausprägungen des Modells in Anhang C möglich, je nachdem welche Infrastruktur gegeben ist. In [Hinckley, 2003] wird vorgeschlagen, alle Geräte über eine WLAN-Infrastruktur miteinander zu vernetzen und ein Gerät als Server für die Gesten-Erkennung zu verwenden. Diese Variante wird auch in diesem Beispiel vorausgesetzt.

Aufbauend auf der Erkennung der Bump-Geste können jetzt die Patterns der einzelnen Kategorien näher spezifiziert werden. Abbildung 4.3 beschreibt das Pattern „Bump To Connect“ und zeigt den Ablauf des Verbindungsaufbaus zwischen zwei

Geräten, nachdem eine Bump-Geste erkannt wurde. Das Gerät, welches sich im Netzwerk auffindbar macht, spielt ein kurzes Geräusch ab. Anschließend wird das Verbindungsprotokoll ausgehandelt.

Bei Erfolg wird ein zusammengesetztes Feedback aus visuellen und akustischen Hinweisen auf beiden Geräten ausgegeben. Im Nichterfolgsfall gibt es ein negatives Feedback. In Abbildung 4.10 findet sich eine statische Beschreibung der verwendeten individuellen Feedbacks und der benötigten *Capabilities*, ansonsten unterliegt die *Connect*-Aktivität einem starren Ablauf, der auch für andere Gesten gleich ablaufen würde. Das Ergebnis der Aktivität ist ein *Channel* vom Typ WLAN, wie oben definiert wurde.

Zwar sieht kein Pattern explizit eine *Select*-Aktivität vor, dennoch ist diese die Grundlage für Patterns mit *Transfer*-Aktivitäten (s. Modell in Abbildung 3.1 bzw. 3.5). Als einfaches Beispiel soll hier der Anwendungsfall aus [Hinckley, 2003] angenommen werden, in dem eine geöffnete Webseite mit einem anderen Benutzer per Bump-Geste geteilt werden soll. Hier ist die *Selection* vom Typ *Command* (vgl. Kapitel 3.3.5), eine Beschreibung des Ablaufes zeigt Abbildung 4.4. Es ist auch möglich, dass ein Nutzer gar nichts auswählt und nur eine leere Anwendung geöffnet hat. Dieser Fall ist in Abbildung 4.5 beschrieben. Mit diesen beiden Definitionen für mögliche *Select*-Aktivitäten lassen sich die drei übrigen Bump-Patterns beschreiben.

Abbildung 4.6 zeigt das Anstoßen eines Datentransfers mittels Bump-Geste. Vorbedingungen sind eine bestehende WLAN-Verbindung und ein ausgewählter *Command*. Wird nun eine synchrone Bump-Geste erkannt, wird der Transfer gestartet. Die Details der Übertragung sind in Abbildung 4.7 dargestellt. Hier wurden einige Schritte, die im Gesamtmodell in Abbildung 3.22 vorgesehen sind, weggelassen. Es wird davon ausgegangen, dass die Übertragung eines *Commands* unmittelbar stattfindet und kein Feedback über den Fortschritt benötigt wird. Zudem gibt zunächst nur das sendende Gerät ein Feedback in Form eines „Teleport“-Sounds aus, um zu signalisieren, dass es den Befehl versendet hat. Auf dem Empfangsgerät ist kein zusätzliches Feedback notwendig, da bereits das Öffnen der Webseite im Browser (die Ausführung des *Command*) ein Feedback an den Nutzer darstellt.

Mittels dieser Beschreibung lassen sich die Patterns „Bump To Give“, „Bump To Take“ und „Bump To Exchange“ beschreiben. Sollte auf einem Gerät eine leere *Selection* vorhanden sein (siehe Abbildung 4.5), so wird höchstens ein „Bump To Ta-

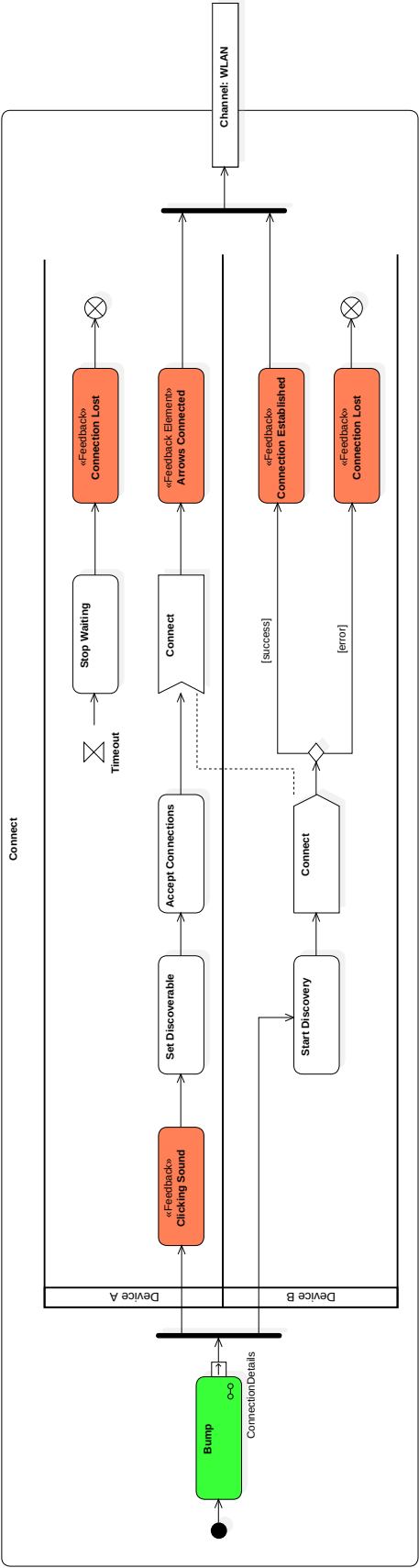
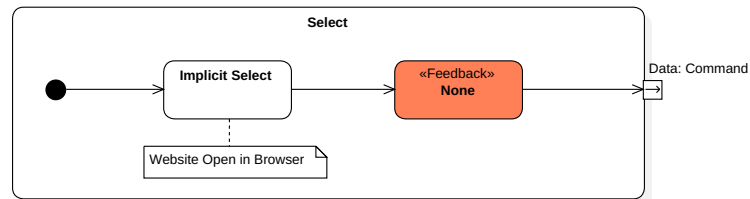
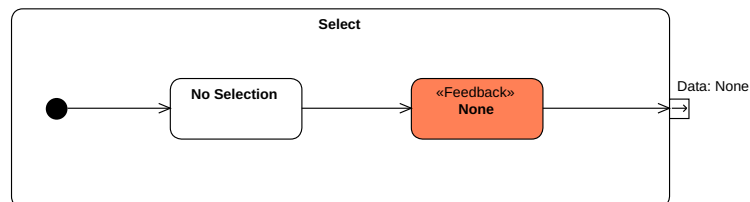


Abbildung 4.3.: Ablauf des Patterns Bump-To-Connect – Verbinden zweier Geräte mittels Bump-Geste.

## 4. Validierung des Modells



**Abbildung 4.4.:** Implizite Auswahl einer geöffneten Webseite im Browser des Nutzers als *Command*.



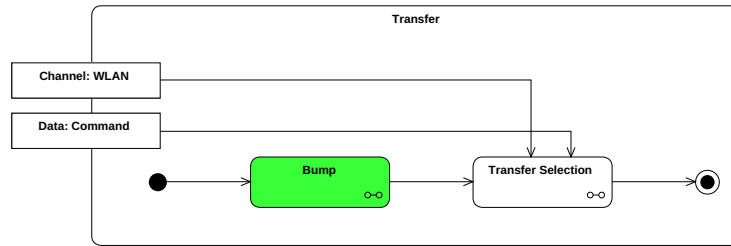
**Abbildung 4.5.:** Keine Auswahl durch den Benutzer.

ke“ durchgeführt, wenn auf dem anderen Gerät der entsprechende *Command* ausgewählt wurde. Das sendende Gerät realisiert dann automatisch ein „Bump To Give“. Ist auf beiden Geräten eine Webseite geöffnet, also auf beiden Geräten eine Selektion vorhanden (siehe Abbildung 4.4), wird ein Austausch von Webseiten durchgeführt, also ein „Bump To Exchange“. Durch eine Definition von Constraints könnte eine genauere Rollenverteilung der „Give“- , „Take“- und „Exchange“-Geräte erreicht werden.

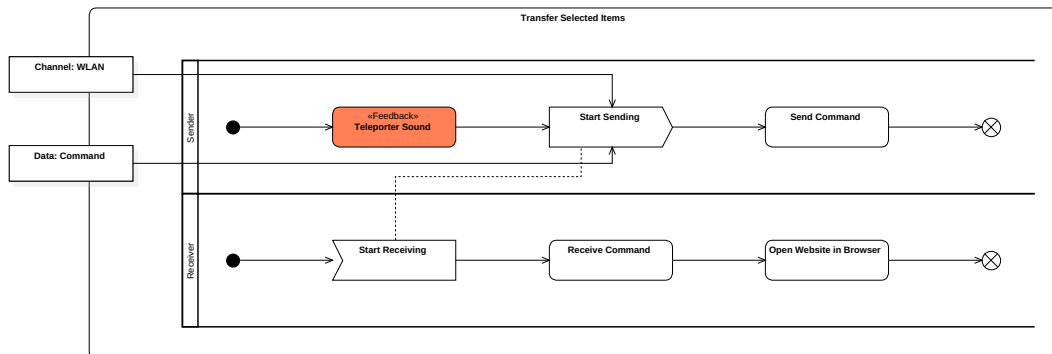
Der Vollständigkeit halber muss nach dem Lebenszyklusmodell (siehe Abbildung 3.1 bzw. 3.5) noch das Trennen der Verbindung definiert werden (siehe Abbildung 4.8). Dazu wird keine Bump-Geste verwendet, da ein „Bump To Disconnect“-Pattern im Pattern-Katalog nicht vorgesehen ist und auch nicht sinnvoll wäre. Somit wird hier einfach davon ausgegangen, dass durch Verlassen des Funkbereiches oder ein eventuelles Timeout-Event irgendwann ein Trennen der Verbindung erfolgt, über das der Nutzer durch ein Feedback informiert wird.

Aus der Gestenbeschreibung in Abbildungen 4.1 und 4.2 ergibt sich das statische Modell in Abbildung 4.9, das die benötigten *Capabilities* berücksichtigt und die erzeugten Events beschreibt. Die verwendeten Feedbacks der verschiedenen Ablaufbeschreibungen wurden in Abbildung 4.10 gesammelt und ebenfalls mit den benötigten *Capabilities* versehen.

Aus den gesammelten Anforderungen an Geräteeigenschaften kann ein abstraktes Gerät modelliert werden, das mindestens notwendig ist, um die Pattern umzusetzen, wie in Abbildung 4.11 zu sehen. Jedes Bump-Pattern benötigt für die Erkennung



**Abbildung 4.6.:** Start des Transfers mittels Bump-Geste.



**Abbildung 4.7.:** Übertragen des *Commands* und Öffnen der Webseite auf dem Empfangsgerät.

der Bump-Geste somit mindestens einen Beschleunigungssensor und muss WLAN-fähig sein (sowohl zum Synchronisieren von *BumpEvents* als auch zum Transfer von Daten). Für die meisten Feedbacks ist Bild- und Tonausgabe wichtig, da oft kombiniertes Feedback eingesetzt wird.

Abbildung 4.12 ergänzt das statische Modell um die verwendeten *Selection*-Typen *Command* und *Empty*, je nachdem ob eine Webseite geöffnet ist oder nicht.

Die statischen Sichten sind nicht extra nach Pattern-Kategorie unterteilt, sondern umfassen jeweils die gesamte Gruppe von Bump-Patterns. Möchte man nur die Anforderungen an ein einzelnes Pattern beschreiben, müssen entsprechend *Feedbacks* und *Capabilities* entfernt werden, die in dem Pattern nicht benötigt werden. Das würde allerdings für die Bump-Patterns keinen großen Unterschied machen, da ähnliche *Capabilities* von allen Patterns vorausgesetzt werden.

Würden statt einer geöffneten Webseite Dateien oder ein Datenstrom übertragen, müsste man nicht das gesamte Modell anpassen, sondern nur die Aktivitäten *Transfer* und *Select* und die dort verwendeten Feedbacks. Somit sind Teile des Modells wie die Gestenerkennung wiederverwendbar, auch Teile der Feedbacks wie bspw. abgespielte Sounds können dort wiederverwendet werden. Durch Erweiterung des statischen Modells kann wiederum darauf geschlossen werden, ob dadurch neue

## 4. Validierung des Modells

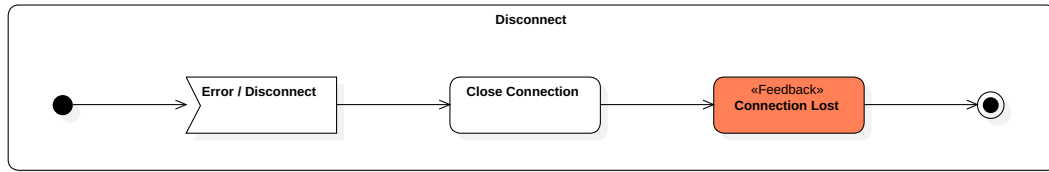


Abbildung 4.8.: Trennen der Verbindung zwischen den Geräten.

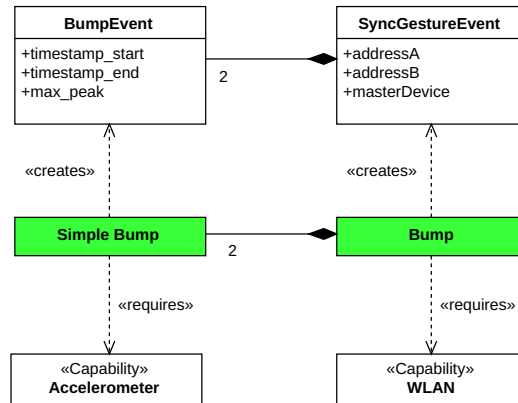


Abbildung 4.9.: Statische Sicht der Erkennung von Bump-Gesten und den dazu benötigten *Capabilities*.

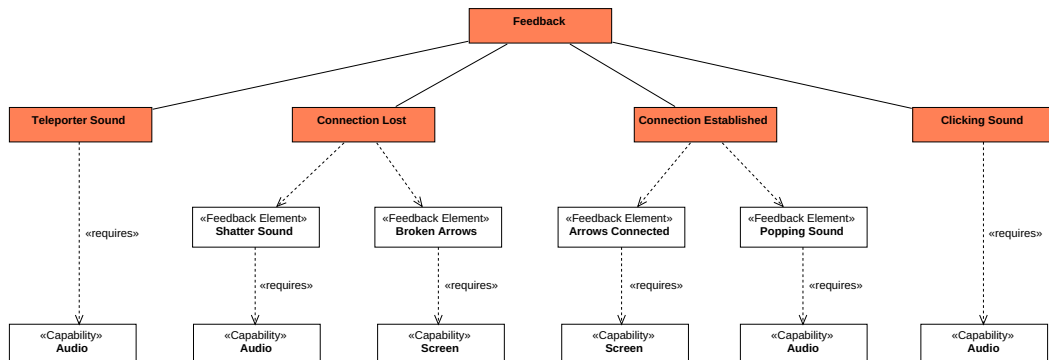


Abbildung 4.10.: Von Bump-Patterns verwendete einfache und zusammengesetzte Feedbacks.

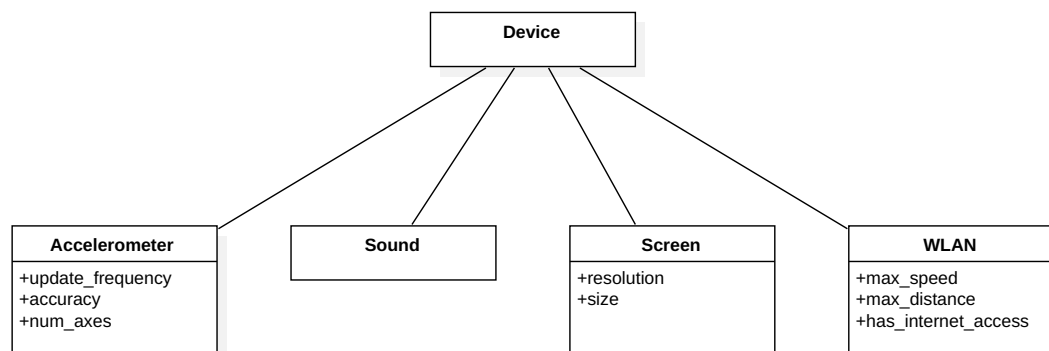


Abbildung 4.11.: Statische Sicht aller benötigten *Capabilities* und deren Zusammenfassung in einem abstrakten Gerät.



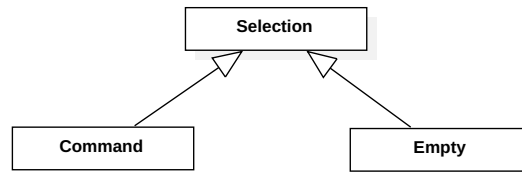


Abbildung 4.12.: Verwendete *Selection*-Typen.

*Capabilities* benötigt werden. Ändert sich das statische Modell in der Hinsicht nicht, kann das Pattern auf den bisher unterstützten Geräten weiterhin eingesetzt werden.

### 4.2.2. Approach-Patterns

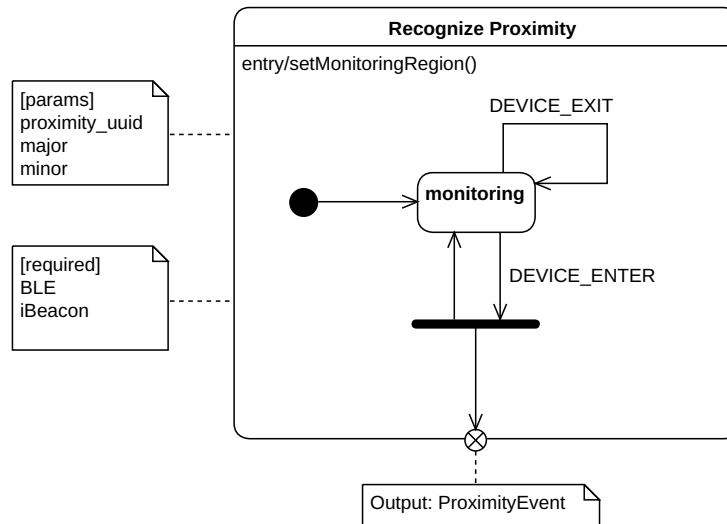
Analog zu den Bump-Patterns wird in diesem Kapitel die Gruppe der Patterns besprochen, die auf der Approach-Geste, also einem Annähern eines Gerätes an ein anderes, basieren. Der SysPlace-Patternkatalog sieht eine Gruppe von vier Patterns vor, die auf dieser Geste basieren (siehe Anhang B):

- Approach To Connect,
- Approach To Give,
- Approach To Take und
- Approach To Extend.

Zudem gibt es das Pattern „Leave To Disconnect“, das sich durch die gleichen Komponenten erfassen lässt, wie später gezeigt wird.

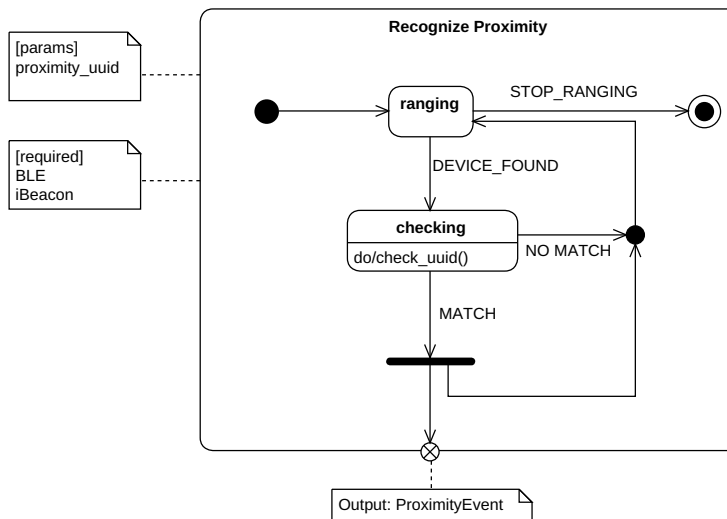
Wie in Kapitel 2 beschrieben gibt es verschiedene Technologien, um Geräte in der Umgebung zu erkennen und ggf. den Abstand zu diesen zu messen. Für die Modellierung der Approach-Patterns wird hier die iBeacon-Technologie verwendet, da diese inzwischen weite Verbreitung findet und das zugrunde liegende BLE in vielen neueren Geräten vorhanden und einfach einzusetzen ist. Alle Beschreibungen lassen sich aber ebenso mit wenig Aufwand auf andere Protokolle wie z. B. Eddystone übertragen.

Abbildung 4.13 zeigt die Erkennung einer Approach-Geste durch sog. *Monitoring* von iBeacons. Jedes iBeacon sendet kontinuierlich ein Signal aus, welches eine UUID sowie eine Major- und Minor-Nummer transportiert (vgl. Kapitel 2.4.1). Beim Monitoring werden diese drei Werte, die zusammen eine sog. *Region* bilden, zur Überwachung im Hintergrund festgelegt. In einem festen Takt wird nun



**Abbildung 4.13.:** Monitoring zum einmaligen Erkennen neuer Geräte bei Annäherung.

vom Betriebssystem mitgeteilt, wenn ein Gerät mit den entsprechenden Werten in den Empfangsbereich ein- oder ausgetreten ist. Es können statt fester Werte auch Werte-Masken verwendet werden, um flexible Regionen zu überwachen. Die Gestererkennung überwacht die in den Parametern definierte Region und reagiert auf Geräte, die neu in den Sendebereich eintreten. Für jedes dieser Geräte wird ein *ProximityEvent* generiert und die Geste somit erfolgreich erkannt. Benötigt werden für die Erkennung BLE sowie eine Implementierung des iBeacon-Protokolls.



**Abbildung 4.14.:** Kontinuierliches Erkennen von Geräten in der Umgebung.

Mittels Monitoring kann erstmal nur erkannt werden, ob spezifische Geräte, die für eine Interaktion in Frage kommen, in den Sendebereich ein- oder austreten. Ab-

Abbildung 4.14 zeigt eine weitere Variante der Approach-Erkennung, basierend auf iBeacon-Ranging. Hier wird nicht im Hintergrund nach Geräten gesucht, sondern ein aktiver Scan durchgeführt. Dieser ist meist zeitlich begrenzt und läuft nicht dauerhaft im Hintergrund, weshalb ein Terminieren der Erkennung durch ein Event wie bspw. ein Timeout vorgesehen ist. Zudem wird hier zwar auf gewisse UUIDs gefiltert (um z. B. iBeacons anderer Hersteller oder Applikationen auszuschließen), die Major- und Minor-Nummern werden aber in diesem Schritt nicht eingeschränkt. Ein Vorteil beim Ranging von iBeacons ist, dass durch das Betriebssystem bereits eine Schätzung des Abstandes (*Proximity*) zum erkannten Gerät vorgenommen wird, üblicherweise in den Abstufungen:

- *immediate*: Abstand von wenigen cm.
- *near*: Abstand üblicherweise weniger als 3 m.
- *far*: Abstand von mehreren Metern.
- *unknown*: Abstand kann nicht ermittelt werden, weil das Signal sehr schwach ist.

Analog zum Monitoring wird bei Erkennen eines Gerätes in der Nähe ein *ProximityEvent* erzeugt.

Um die Gesten für einen bestimmten Anwendungskontext nutzen zu können, müssen entsprechend dem Modell in Abbildung 3.6 auf der Erkennung aufbauend noch Constraints definiert werden, um die Gestenerkennung anzupassen. Abbildung 4.15 zeigt zwei Constraints, die die Major-Nummer sowie den geschätzten Abstand einschränken. Während die UUID oft zum Identifizieren von Firmen oder Herstellern verwendet wird, kann über die Major-Nummer z. B. festgelegt werden, welcher Service auf dem erkannten Gerät bereitgestellt wird. So kann entschieden werden, ob eine Interaktion möglich ist und eine Annäherungsgeste vorliegt.

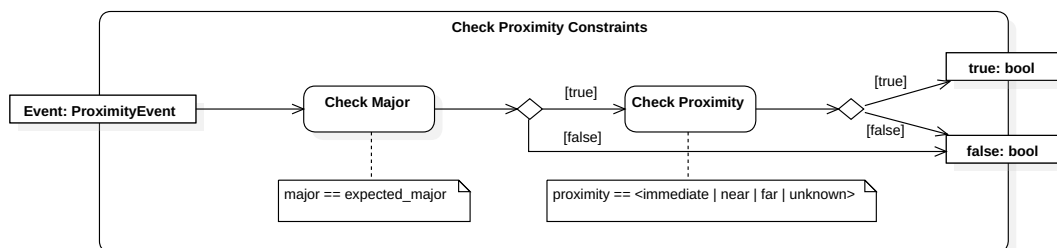
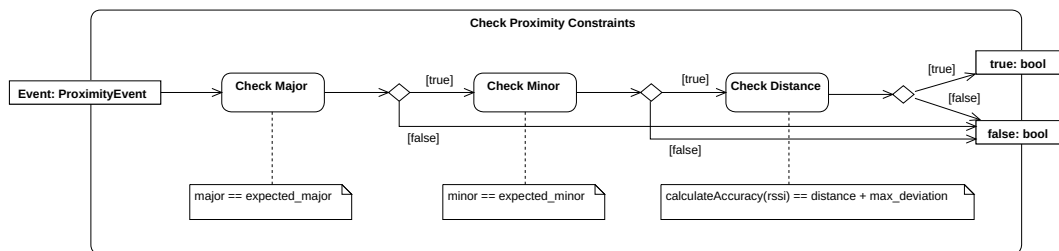


Abbildung 4.15.: Constraints für Major-Nummer und Proximity-Zone prüfen.

## 4. Validierung des Modells

Abbildung 4.16 zeigt eine noch detailliertere Variante der Constraint-Prüfung, in der sowohl Major- als auch Minor-Nummer geprüft werden und eine zusätzlich aus der gemessenen Sendeleistung des erkannten iBeacon-Gerätes (*rssi*) eine absolute Distanz in cm abgeleitet werden kann. Diese Funktion steht in den meisten iBeacon-Implementierungen zur Verfügung, hat allerdings eine schwankende Genauigkeit, z. B. wegen hoher Sendelast auf dem verwendeten ISM-Frequenzband. Daher ist hier zusätzlich der Parameter *max\_deviation* vorgesehen, um einen Toleranzbereich für diese Schwankungen zu definieren.



**Abbildung 4.16.:** Constraints für Major-Nummer, Minor-Nummer und Distanz prüfen.

Die Kombination des Ranging-Ansatzes mit einem solchen Constraint Check erlaubt es, kontinuierlich auf die Abstandsänderung zu erkannten Geräten in der Umgebung zu reagieren und Interaktionen mittels dieser Geste zu gestalten. Soll nur die reine Anwesenheit von Geräten einmalig beim Ein- oder Austritt erkannt werden, reicht die Monitoring-Variante aus.

Abbildung 4.17 zeigt, wie mittels der Approach-Geste und dem erzeugten Proximity-Event die Geste „Approach To Connect“ beschrieben werden kann. Da es sich hierbei nicht um eine synchrone Geste handelt, muss kein weiterer Vergleich von Events vorgenommen werden. Allerdings kommt hier die Variante des Modells aus Abbildung 3.18 zum Einsatz, in der eine einfache Geste den Verbindungsaufbau initiiert und das andere Gerät die Möglichkeit zum Ablehnen oder Annehmen der Verbindung hat. Da iBeacon auf Bluetooth basiert, wird als *Channel* hier eine Bluetooth-Verbindung angenommen, die die Geräte direkt miteinander verbindet. Dazu muss allerdings ein zusätzliches Bluetooth-Pairing durchgeführt werden (siehe Kapitel 2.4.1).

Die Gesten „Approach To Give“, „Approach To Take“ und „Approach To Extend“ würden analog zu den Bump-Patterns aus Kapitel 4.2.1 beschrieben und werden hier nicht noch einmal im Detail besprochen. Eine Besonderheit ist hier die Geste „Approach To Extend“, da zwei Geräte gekoppelt werden. In dem Fall wäre die

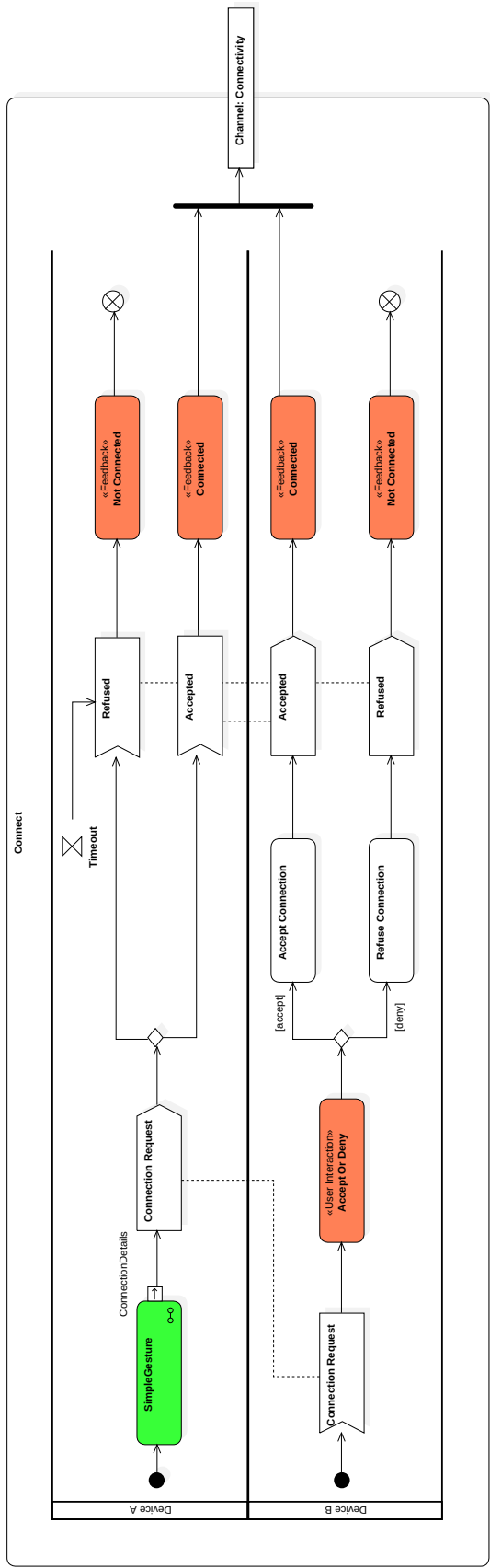


Abbildung 4.17.: Verbinden zweier Geräte mittels Approach-Geste.

## 4. Validierung des Modells

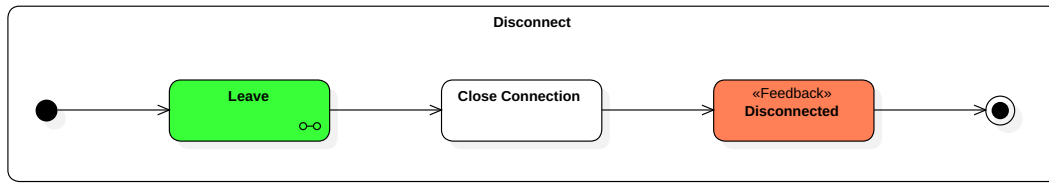


Abbildung 4.18.: Trennen der Verbindung mittels Leave-Geste.

*Selection* zumeist vom Typ *Stream* und es würde eine komplexere, kontinuierliche Datenverarbeitung stattfinden und nicht bloß das Ausführen eines Befehls. Denkbar ist auch das Aufteilen eines Bildes in mehrere Teile. Dazu muss eine Berechnungslogik definiert werden, die festlegt, in welchen Verhältnissen das Bild aufgeteilt werden soll.

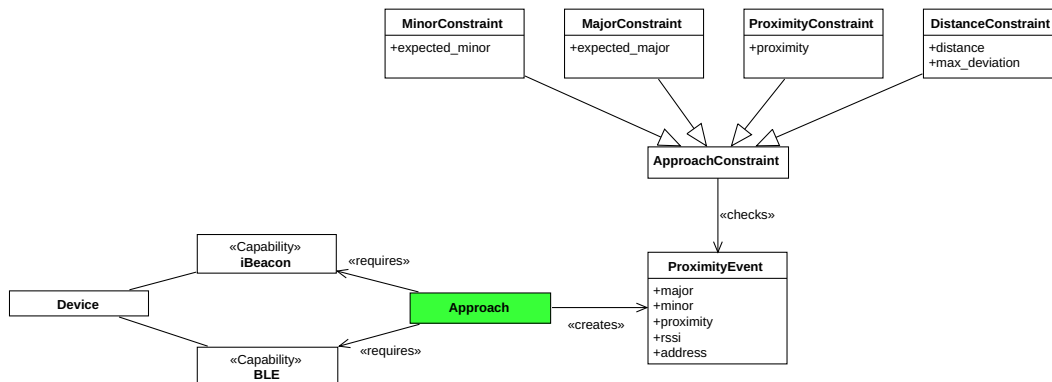


Abbildung 4.19.: Domänenmodell der Approach-Gestenerkennung.

Wie eingangs erwähnt, lässt sich neben den Approach-Pattern auch das Pattern „Leave To Disconnect“ mit der Approach-Gestenerkennung beschreiben. Abbildung 4.13 zeigt das Monitoring von iBeacon-Geräten in der Umgebung. Wenn das Eintreten eines Gerätes (*DEVICE\_ENTER*) erkannt wird, wird das *ProximityEvent* erzeugt; verlässt ein Gerät die Umgebung, passiert nichts (*DEVICE\_EXIT*). Vertauscht man die beiden Events, erhält man eine Abwandlung zur Erkennung von Leave-Gesten, die ein *ProximityEvent* erzeugen, wenn ein Gerät den Sendebereich verlassen hat. Bestand eine Verbindung zu diesem Gerät (was über einen Constraint Check auf die Adresse des *ProximityEvents* geprüft werden könnte), kann der in Abbildung 4.18 beschriebene Verbindungsabbau begonnen werden.

Eine statische Sicht auf die Erkennung der Approach-Geste, die benötigten *Capabilities* und die verwendeten Constraints findet sich in Abbildung 4.19. Da hier nicht auf Feedback eingegangen wurde, fehlt dieser Teil im Modell. Für die Gestaltung einer Applikation auf Basis der hier beschriebenen Gestenerkennung müssten Feed-

backs und Nutzer-Interaktionen in den Ablaufdiagrammen sowie in der statischen Sicht des Modells noch entsprechend ergänzt werden.

### 4.3. Diskussion der Anwendbarkeit für weitere Patterns

Im SysPlace-Patternkatalog sind in der aktuellen Version knapp 30 Patterns erfasst (vgl. Anhang B). Das Ziel des Modells ist es, all diese Patterns beschreiben zu können. In den Kapiteln 4.2.1 und 4.2.2 wurde gezeigt, dass das für Patterns basierend auf Approach-, Leave- und Bump-Gesten möglich ist.

Ein anschauliches Beispiel für die Erkennung einer Swipe-Geste wird in Kapitel 3.3.2 gezeigt. Aufbauend darauf wird in Kapitel 3.3.3 die Erkennung einer synchronen Stitch-Geste exemplarisch beschrieben. Würde man diese Gestenerkennung um Details entsprechend dem Applikationslebenszyklus aus Kapitel 3.3.1 ergänzen, würde man vollständige Beschreibungen für die Patterns

- Stitch To Give,
- Stitch To Take,
- Stitch To Extend und
- Stitch To Connect

erhalten. Von den gegebenen Beispielen ausgehend kann auf die Anwendbarkeit des Modells für die übrigen Patterns geschlossen werden.

Das Pattern „Swipe To Give“ basiert auf einer einfachen Swipe-Geste in Richtung des Bildschirmrandes und kann durch die Swipe-Erkennung und einen *SwipeConstraint*, der die Richtung einschränkt, realisiert werden. Die Stitch-Erkennung kann abgewandelt werden, um das Pattern „Extend With Two Fingers“ zu beschreiben. Dazu müssen lediglich zwei Swipe-Gesten in entgegengesetzter Richtung, jeweils in Richtung des Bildschirmrandes, erkannt werden, statt wie beim *Stitch Event Matching* in gleicher Richtung.

Kapitel 4.2.2 zeigt die Erfassung von Patterns, die auf einer Annäherungsgeste basieren. Das Erkennen von Geräten in der Umgebung spielt auch in den Patterns „Picking Up An Object“, „Grab An Object“, „Grab A Part“ und „Appose“ eine Rolle. Hier liegt jeweils eine synchrone Geste zugrunde, in der entweder ein Gerät eine Approach-Geste und das andere Gerät eine Auflage-Geste erkennt bzw. beide

Geräte gleichzeitig eine Annäherung wahrnehmen. Ein Auflegen von Geräten kann bspw. über Infrarot-Touchscreens, Kameras oder NFC erkannt und als Sensor-Event in einer einfachen Gestenerkennung verwendet werden. Durch die Kombination mit der bereits erfassten Approach-Geste und einem entsprechenden *Event Matching* können Patterns dieser Art erfasst werden.

Die Erkennung eines Beschleunigungsmusters bildet den Ausgangspunkt für die Bump-Erkennung in Kapitel 4.2.1. Eine Abwandlung des erkannten Musters ermöglicht es, die Patterns „Tennis“, „Frisbee“, „Dump“ und „Nudge“ zu erfassen, wobei letzteres als synchrone Geste erfasst werden muss, bei der das Zielgerät die Berührung des Bildschirms erkennt. Auch das Pattern „Shake Well To Connect“ kann als synchrone Geste zweier einfacher Shake-Gesten dargestellt werden, die wiederum auf einem einfachen Bewegungsmuster mit alternierenden Spitzen basieren. Allerdings könnte es sich schwierig gestalten, eine Beschreibung komplexer Bewegungsmuster wie einem Frisbee-Wurf zu erfassen, da viele Änderungen in der Beschleunigung viele Zustände und Transitionen im Modell erzeugen.

Um die Patterns „Give Through Body“ und „Exchange Through Body“ umzusetzen, reicht das Erkennen einer einfachen synchronen Geste, bei der gleichzeitige Bildschirmberührungen erkannt werden, ähnlich dem SyncTap-System, das in Kapitel 3.3.3 vorgestellt wurde. Da allerdings der Nutzer als „Verbindung“ zwischen den Geräten fungiert, muss das Trennen der Verbindung und Beenden von Transfers ausgelöst werden, wenn er einen oder beide Bildschirme wieder loslässt. Auch das ist durch das Modell problemlos möglich, indem die *Disconnect*-Aktivität mit einem entsprechenden Auslöser modelliert wird.

Das Pattern „World in Miniature“ ist schwieriger durch das Modell zu beschreiben. Hier gibt es keine zentrale Geste, wie es bei allen anderen Patterns der Fall ist. Es wird eine Verbindung vorausgesetzt und beschrieben, welcher Art die ausgetauschten Daten sind, nicht aber, in welcher Form der Nutzer dazu mit dem System interagieren kann.

Zusammenfassend lässt sich sagen, dass

- alle Patterns (abgesehen von „World in Miniature“) durch das Modell beschrieben werden können, aber
- komplexe Gestenmuster oder Gesten mit einer hohen Varianz in der Ausführung eventuell schwierig durch ein Zustandsdiagramm zu erfassen sind und



- bei der Beschreibung neuer Patterns Komponenten bereits erfasster Patterns wiederverwendet werden können (bspw. Approach → Appose).

#### 4.4. Ansatz zur Entwicklung eines Frameworks

Abschließend soll in diesem Kapitel kurz umrissen werden, inwieweit das Modell die Entwicklung eines Frameworks für Multiscreen-Interaktionen unterstützen kann. Dadurch wird eine Brücke zu den weiterführenden Arbeiten und der weiteren Arbeit an dem Modell im Projekt SysPlace geschlagen.

Ein Framework sollte die individuelle Gestaltung von Multiscreen-Interaktionen auf einer bestimmten Plattform unterstützen. Es soll Entwickler in die Lage versetzen, die individuelle Gestaltung von Interaktions- und Feedback-Aktivitäten sowie der Erkennung von Gesten zu implementieren, ohne die Kommunikation der verschiedenen Komponenten entlang der erarbeiteten Abläufe selbst gewährleisten zu müssen. Die Implementierung sämtlicher Aktivitäten, die einheitlich durchführbar sind (wie z. B. die Übertragung von Datenpaketen, das Aufbauen von Verbindungen oder das Auslösen von Systemfeedbacks) können in das Framework verlagert werden.

Der Entwickler implementiert Komponenten wie z. B. einfache oder zusammengesetzte Feedbacks, Gesten-Erkennungen oder Constraints anhand definierter Schnittstellen und registriert sie beim Framework, damit sie an der entsprechenden Stelle im Ablauf per Callback aufgerufen werden. Dadurch werden Komponenten wiederverwendbar und das Anpassen einer Multiscreen-Anwendung ist im Idealfall nur ein Konfigurationsschritt.

Für die Gestenerkennung könnten Standardimplementierungen bereitgestellt werden, die eine einfache Umsetzung der in der Literatur vorgeschlagenen Algorithmen, wie bspw. für den Bump, bieten. Ferner könnte es Standardfeedbacks geben, sodass ein schnelles Prototyping von Anwendungen möglich ist, weil nicht erst alle Komponenten individuell erstellt und mit einer Minimalimplementierung versehen werden müssen.

Eine prototypische Framework-Implementierung wurde im Rahmen des Projektes SysPlace bereits durchgeführt und hatte eine Android-Applikation zum Ergebnis, in der die Gesten

- Shake,

#### 4. Validierung des Modells

---

- Bump,
- Stitch und
- Swipe

mittels einer einheitlich aufrufbaren Komponente zur Gestenerkennung realisiert wurden. Als Übertragungskanal wurden sowohl Bluetooth als auch Wi-Fi Direct in Komponenten mit gemeinsamer Schnittstelle implementiert. Dadurch war es möglich, eine Applikation mittels einfacher Konfiguration zu erstellen, in der eine beliebige Kombination aus Geste und Übertragungskanal den Transfer eines Bildes zwischen zwei Geräten ermöglichte.

## Kapitel 5

# Zusammenfassung und Ausblick

### 5.1. Zusammenfassung der Ergebnisse

Ausgehend von einer Analyse verschiedener Publikationen, die Multiscreen-Szenarien theoretisch entwerfen und prototypisch umsetzen, ist das Ergebnis dieser Arbeit ein Modellierungskonzept, mit dem diese heterogenen Ansätze einheitlich beschrieben werden können. Unter Einbeziehung der in den Grundlagen vorgestellten Technologien können verschiedene Aspekte von Multiscreen-Szenarien durch das Modell beschrieben werden:

- Erkennen von Nutzergesten,
- Aufbauen und Trennen von Verbindungen, basierend auf Gesten,
- Übertragung von Daten, basierend auf Gesten,
- System-Feedback und Nutzerinteraktionen und
- Beziehungen zwischen den einzelnen Aktivitäten im Lebenszyklus einer Multiscreen-Anwendung.

Für die meisten Patterns ließ sich zeigen, dass eine Beschreibung durch das Modell anhand einer Abwandlung oder Kombination bereits exemplarisch vorgenommener Modellierungen möglich ist.

Allerdings hat sich auch gezeigt, dass sich die Modellierung einer kleinen Zahl von Patterns, die eine komplexere Nutzergeste vorsehen, eventuell schwieriger gestaltet. Für diese Fälle ist eventuell eine Erweiterung des Modells um eine spezifischere Möglichkeit der Gesten-Modellierung notwendig, worauf in weiterführenden Arbeiten eingegangen werden sollte. Zusätzlich wurde ein Pattern identifiziert, das

durch das Modell nur schwer beschrieben werden kann, da es eine von den übrigen Patterns abweichende Struktur aufweist.

Ergänzt wurden die theoretischen Ergebnisse noch um einen skizzenhaften Ansatz zur Entwicklung eines Frameworks, das eine einfache Realisierung von Multiscreen-Anwendungen technisch unterstützen könnte.

### 5.2. Weiterführende Arbeiten

Der nächste Schritt im Projekt SysPlace ist der Einsatz des Modells, um die technische Realisierung der Patterns im SysPlace-Patternkataloges entsprechend der Zielsetzung einheitlich und detailliert zu beschreiben. Anknüpfende Arbeiten könnten zudem Erweiterungen des Modells sowie Referenz-Implementierungen sein, worauf im folgenden eingegangen wird.

Auf der Modellierungsebene sind weiterführende Arbeiten denkbar, um andere Forschung zur Formalisierung von Nutzerschnittstellen oder der Erkennung von Gesten miteinzubeziehen und das Modell so an einigen Stellen mit einem höheren Detailgrad auszustatten. Bisher ist keine abstrakte Beschreibung von Nutzeroberflächen oder visuellem Feedback vorgesehen. Dazu könnte das Modell um eine UIDL<sup>1</sup> erweitert werden, wie z. B. der User Interface Extensible Markup Language (USIXML), womit Struktur und Verhalten von Nutzeroberflächen abstrakt beschrieben und in konkrete Implementierungen überführt werden können [Limbourg u. a., 2004].

Auch für die Gestenerkennung ist eine Integration von bestehenden Formalisierungskonzepten denkbar. Ein Beispiel hierfür wäre die von Kammer und Kollegen entworfene Gesture Formalization for Multi-touch (GeForMT). Dabei handelt es sich um eine formale Sprache, die speziell auf die Definition von Multitouch-Gesten zugeschnitten ist und auch komplexe Gesten präzise erfassen kann [Kammer u. a., 2010]. Der Vorteil wäre neben einer deutlich einfacheren Dokumentation der Gestenerkennung die Möglichkeit, die Beschreibungen direkt in lauffähigen Code zu übersetzen, da eine Implementierung als JavaScript-Bibliothek bereits vorliegt

---

<sup>1</sup>User Interface Description Languages (UIDLs) ermöglichen die formale Beschreibung von Nutzeroberflächen, meist unter Verwendung deklarativer Auszeichnungssprachen wie z. B. XML. Oft wird dabei ein mehrschichtiges Modell verwendet, in dem aus abstrakten, plattformunabhängigen Beschreibungen konkrete Implementierungen abgeleitet werden.

und neben einer vollständigen Grammatikbeschreibung frei verfügbar ist [Kammer, 2013].

Die in der Analyse und der Validierung vorgestellten Beispiele zur Gestenerkennung definieren verschiedene Parameter, die für die korrekte Erkennung von Gesten geprüft werden müssen. Um diese Parameter mit konkreten Werten bzw. möglichen Wertebereichen zu versehen, sind weiterführende Evaluationen denkbar. Im Projekt SysPlace wurde bereits eine empirische Ermittlung eines Wertebereiches für den Schwellwert der Bump-Gestenerkennung im Rahmen einer Bachelorarbeit vorgenommen. Für andere Gesten sind ähnliche Arbeiten denkbar, wie z. B. für Distanzwerte von Approach-Gesten oder für Länge, Dauer und Richtung von Swipe-Gesten.

Auf der technischen Ebene bildet das Modell die Grundlage zur Entwicklung eines Frameworks, da verschiedene Komponenten und deren Zusammenspiel im Lebenszyklus einer Multiscreen-Anwendung bereits identifiziert wurden und einfach in Software-Komponenten und Ablaufstrukturen übertragen werden können. Im Rahmen des Projektes SysPlace ist vorgesehen, eine prototypische Realisierung für die Android-Plattform vorzunehmen.



# Abkürzungsverzeichnis

<b>API</b>	Application programming interface
<b>AR</b>	Augmented Reality
<b>BLE</b>	Bluetooth Low Energy
<b>GeForMT</b>	Gesture Formalization for Multi-touch
<b>GUI</b>	Graphical User Interface
<b>ISM</b>	Industrial, Scientific and Medical
<b>KMU</b>	kleine und mittelständische Unternehmen
<b>LAN</b>	Local Area Network
<b>MDE</b>	multi-display environment
<b>NFC</b>	Near Field Communication
<b>OMT</b>	Object Modeling Technique
<b>PDA</b>	Personal Digital Assistant
<b>RFID</b>	radio-frequency identification
<b>SIG</b>	Bluetooth Special Interest Group
<b>TCP</b>	Transmission Control Protocol
<b>UDP</b>	User Datagram Protocol
<b>UIDL</b>	User Interface Description Language
<b>UML</b>	Unified Modeling Language
<b>URL</b>	Uniform Resource Locator
<b>USXML</b>	User Interface Extensible Markup Language
<b>UUID</b>	Universally Unique Identifier
<b>VR</b>	Virtual Reality
<b>WLAN</b>	Wireless Local Area Network
<b>XML</b>	Extensible Markup Language





# Tabellenverzeichnis

2.1. Gängige WLAN-Standards. Quelle: [Kurose und Ross, 2014, 562]. .	13
--	----



# Abbildungsverzeichnis

2.1.	Die vier Geräteklassen Smartphone, PC/Laptop, Tablet, (Smart) TV. Quelle: [Google, 2012]. . . . .	8
2.2.	Die vier Geräteklassen und deren übliche Screengröße in Relation. Quelle: [Nagel und Fischer, 2013]. . . . .	9
2.3.	WLAN im Infrastrukturmodus. . . . .	13
2.4.	WLAN im Ad-Hoc-Modus. . . . .	14
2.5.	Piconetz aus einem Master und der Maximalzahl von sieben Slaves. Angelehnt an [Fuchß, 2009, 125]. . . . .	15
2.6.	Nutzung von UUID, Major- und Minor-Nummer zur Realisierung einer Lokalisierung mittels iBeacon. Quelle: [Apple, 2014]. . . . .	16
2.7.	Die drei Betriebsmodi von NFC. Quelle: [NFC Forum, 2015b]. . . . .	19
2.8.	Das Standardkoordinatensystem für Sensoren mobiler Geräte. Quel- le: [Android, 2015c]. . . . .	19
2.9.	Azimuth, Pitch und Roll bei mobilen Geräten. Quelle: [MathWorks, 2014]. . . . .	21
3.1.	Rahmenmodell für Multiscreen-Interaktion. . . . .	31
3.2.	Smart-Its Friends Sender / Empfänger. Quelle: [Holmquist u. a., 2001]. . . . .	32
3.3.	Wurf-Geste zur Interaktion zwischen Smartphone und Smart TV. Quelle: [Dachselt und Buchholz, 2009]. . . . .	33
3.4.	Stitch-Geste zur Übertragung von Bildern. Quelle: [Hinckley u. a., 2004]. . . . .	33
3.5.	Rahmenmodell mit zusammengefassten <i>Connect</i> - und <i>Transfer</i> -Ak- tivitäten. . . . .	34
3.6.	Modell zur Erkennung einfacher Gesten. . . . .	35
3.7.	Algorithmus zur Erkennung einer Swipe-Geste anhand von Touch- Events. . . . .	37
3.8.	Vergleich der erkannten Swipe-Geste mit vordefinierten Einschrän- kungen ( <i>Constraints</i> ). . . . .	37
3.9.	Drei Ausprägungen der Pinch-Geste. Quelle: [Nielsen u. a., 2014]. . . . .	38
3.10.	Kommunikationsprotokoll zur Erkennung synchroner Gesten. . . . .	40
3.11.	Vergleich von <i>GestureEvents</i> zur Erkennung synchroner Gesten. . . . .	41

3.12. Zusammenhang zwischen Swipe-Events und Stitch-Erkennung. Quelle: [Hinckley u. a., 2004]. . . . .	41
3.13. Vergleich zweier Swipe-Events zur Erkennung der synchronen Stitch-Geste. . . . .	42
3.14. Typische Beschleunigungsmuster der Bump-Geste. Quelle: [Hinckley, 2003]. . . . .	43
3.15. Erkennung einer synchronen Geste basierend auf Buttons im System SyncTap. Quelle: [Rekimoto, 2004]. . . . .	44
3.16. Aufbauen einer Verbindung zwischen zwei Geräten mittels synchroner Geste. . . . .	46
3.17. Optionsmenü als visuelles Feedback nach erfolgreicher Stitch-Geste. Quelle: [Hinckley u. a., 2004]. . . . .	49
3.18. Aufbauen einer Verbindung zwischen zwei Geräten mittels einfacher Geste. . . . .	50
3.19. Ablauf der Aktivität <i>Select</i> . . . . .	52
3.20. Auswahl mehrerer Dateien im System StitchMaster. (a) Auswahl der Dateien mittels Lasso-Geste. (b) Hervorhebung der Auswahl durch das System. Quelle: [Hinckley u. a., 2004]. . . . .	53
3.21. Beginn des Transfers mit Vorbedingungen und Auslösern. . . . .	56
3.22. Ablauf eines Datentransfers in der Aktivität <i>Transfer Selected Items</i> . . . . .	57
3.23. Trennen der Verbindung in der Aktivität <i>Disconnect</i> . . . . .	60
3.24. Domänenmodell für die Gestenerkennung. . . . .	64
3.25. Domänenmodell der Geräteeigenschaften ( <i>Capabilities</i> ). . . . .	65
3.26. Aufbau von <i>User Interactions</i> und <i>Feedback</i> mit Bezug zu <i>Capabilities</i> . . . . .	66
3.27. Domänenmodell für die <i>Selection</i> . . . . .	66
4.1. Erkennung einer Bump-Geste entsprechend der <i>Gesture Detection</i> in Abbildung 3.6. . . . .	74
4.2. Erkennung einer synchronen Bump-Geste entsprechend dem Modell in Abbildung 3.10. . . . .	75
4.3. Ablauf des Patterns Bump-To-Connect – Verbinden zweier Geräte mittels Bump-Geste. . . . .	77
4.4. Implizite Auswahl einer geöffneten Webseite im Browser des Nutzers als <i>Command</i> . . . . .	78
4.5. Keine Auswahl durch den Benutzer. . . . .	78
4.6. Start des Transfers mittels Bump-Geste. . . . .	79
4.7. Übertragen des <i>Commands</i> und Öffnen der Webseite auf dem Empfangsgerät. . . . .	79
4.8. Trennen der Verbindung zwischen den Geräten. . . . .	80
4.9. Statische Sicht der Erkennung von Bump-Gesten und den dazu benötigten <i>Capabilities</i> . . . . .	80
4.10. Von Bump-Patterns verwendete einfache und zusammengesetzte Feedbacks. . . . .	80

---

4.11. Statische Sicht aller benötigten <i>Capabilities</i> und deren Zusammenfassung in einem abstrakten Gerät. . . . .	80
4.12. Verwendete <i>Selection</i> -Typen. . . . .	81
4.13. Monitoring zum einmaligen Erkennen neuer Geräte bei Annäherung. . . . .	82
4.14. Kontinuierliches Erkennen von Geräten in der Umgebung. . . . .	82
4.15. Constraints für Major-Nummer und Proximity-Zone prüfen. . . . .	83
4.16. Constraints für Major-Nummer, Minor-Nummer und Distanz prüfen. . . . .	84
4.17. Verbinden zweier Geräte mittels Approach-Geste. . . . .	85
4.18. Trennen der Verbindung mittels Leave-Geste. . . . .	86
4.19. Domänenmodell der Approach-Gestenerkennung. . . . .	86
 C.1. Erkennung einer synchronen Geste mit einem Gerät als Erkennungsserver. . . . .	 xl
C.2. Parallele Gestenerkennung auf jedem Gerät. . . . .	xli



# Literaturverzeichnis

- [Alexander u. a. 1977] ALEXANDER, Christopher ; ISHIKAWA, Sara ; SILVERSTEIN, Murray: *A Pattern Language: Towns, Buildings, Construction*. New York : Oxford University Press, August 1977
- [Android 2015a] ANDROID: *Motion Sensors*. 2015. – URL [http://developer.android.com/guide/topics/sensors/sensors\\_motion.html](http://developer.android.com/guide/topics/sensors/sensors_motion.html). – Zugriffsdatum: 24.09.2015
- [Android 2015b] ANDROID: *Position Sensors*. 2015. – URL [http://developer.android.com/guide/topics/sensors/sensors\\_position.html](http://developer.android.com/guide/topics/sensors/sensors_position.html). – Zugriffsdatum: 24.09.2015
- [Android 2015c] ANDROID: *Sensors Overview*. 2015. – URL [http://developer.android.com/guide/topics/sensors/sensors\\_overview.html](http://developer.android.com/guide/topics/sensors/sensors_overview.html). – Zugriffsdatum: 24.09.2015
- [Apple 2010] APPLE: *Apple Sells Three Million iPads in 80 Days*. 2010. – URL <http://www.apple.com/pr/library/2010/06/22Apple-Sells-Three-Million-iPads-in-80-Days.html>. – Zugriffsdatum: 18.09.2015
- [Apple 2014] APPLE: *Getting Started with iBeacon*. 2014. – URL <https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf>. – Zugriffsdatum: 23.09.2015
- [Aumi u. a. 2013] AUMI, Md Tanvir I. ; GUPTA, Sidhant ; GOEL, Mayank ; LARSON, Eric ; PATEL, Shwetak: DopLink: Using the Doppler Effect for Multi-device Interaction. In: *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. New York, NY, USA : ACM, 2013 (UbiComp '13), S. 583–586
- [Banga und Weinhold 2014] BANGA, Cameron ; WEINHOLD, Josh: *Essential Mobile Interaction Design: Perfecting Interface Design in Mobile Apps*. 1st. Addison-Wesley Professional, 2014
- [Borchers 2000] BORCHERS, Jan O.: A Pattern Approach to Interaction Design. In: *Proceedings of the 3rd Conference on Designing Interactive Systems*:

- Processes, Practices, Methods, and Techniques*. New York, NY, USA : ACM, 2000 (DIS '00), S. 369–378
- [Chen u. a. 2014] CHEN, Xiang ' ; GROSSMAN, Tovi ; WIGDOR, Daniel J. ; FITZMAURICE, George: Duet: Exploring Joint Interactions on a Smart Phone and a Smart Watch. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA : ACM, 2014 (CHI '14), S. 159–168
- [Cho u. a. 2007] CHO, Sung-Jung ; MURRAY-SMITH, Roderick ; KIM, Yeun-Bae: Multi-context Photo Browsing on Mobile Devices Based on Tilt Dynamics. In: *Proceedings of the 9th International Conference on Human Computer Interaction with Mobile Devices and Services*. New York, NY, USA : ACM, 2007 (MobileHCI '07), S. 190–197
- [Dachselt und Buchholz 2009] DACHSELT, Raimund ; BUCHHOLZ, Robert: Natural Throw and Tilt Interaction Between Mobile Phones and Distant Displays. In: *CHI '09 Extended Abstracts on Human Factors in Computing Systems*. New York, NY, USA : ACM, 2009 (CHI EA '09), S. 3253–3258
- [Echtler und Butz 2012] ECHTLER, Florian ; BUTZ, Andreas: GISpL: Gestures Made Easy. In: *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction*. New York, NY, USA : ACM, 2012 (TEI '12), S. 233–240
- [Fuchß2009] FUCHSS, Thomas: *Mobile Computing*. München : Carl Hanser Verlag GmbH & Co. KG, 2009
- [Gamma u. a. 1995] GAMMA, Erich ; HELM, Richard ; JOHNSON, Ralph ; VLISSIDES, John: *Design Patterns: Elements of Reusable Object-oriented Software*. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 1995
- [Google 2012] GOOGLE: *The New Multi-screen World: Understanding Cross-platform Consumer Behaviour*. 2012. – URL [http://think.withgoogle.com/databoard/media/pdfs/the-new-multi-screen-world-study\\_research-studies.pdf](http://think.withgoogle.com/databoard/media/pdfs/the-new-multi-screen-world-study_research-studies.pdf). – Zugriffsdatum: 18.09.2015
- [Google 2015] GOOGLE: *Getting Started with iBeacon*. 2015. – URL <https://developers.google.com/beacons/overview>. – Zugriffsdatum: 23.09.2015
- [Hamilton und Wigdor 2014] HAMILTON, Peter ; WIGDOR, Daniel J.: Conductor: Enabling and Understanding Cross-device Interaction. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA : ACM, 2014 (CHI '14), S. 2773–2782



- [Hansen 2015] HANSEN, Christopher J.: Internetworking with Bluetooth Low Energy. In: *GetMobile: Mobile Comp. and Comm.* 19 (2015), August, Nr. 2, S. 34–38. – ISSN 2375-0529
- [Hinckley 2003] HINCKLEY, Ken: Synchronous Gestures for Multiple Persons and Computers. In: *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology*. New York, NY, USA : ACM, 2003 (UIST '03), S. 149–158
- [Hinckley u. a. 2004] HINCKLEY, Ken ; RAMOS, Gonzalo ; GUIMBRETIERE, Francois ; BAUDISCH, Patrick ; SMITH, Marc: Stitching: Pen Gestures That Span Multiple Displays. In: *Proceedings of the Working Conference on Advanced Visual Interfaces*. New York, NY, USA : ACM, 2004 (AVI '04), S. 23–31
- [Holmquist u. a. 2001] HOLMQUIST, Lars E. ; MATTERN, Friedemann ; SCHIELE, Bernt ; ALAHUHTA, Petteri ; BEIGL, Michael ; GELLERSEN, Hans-Werner: Smart-Its Friends: A Technique for Users to Easily Establish Connections Between Smart Artefacts. In: *Proceedings of the 3rd International Conference on Ubiquitous Computing*. London, UK, UK : Springer-Verlag, 2001 (UbiComp '01), S. 116–122
- [Izadi u. a. 2003] IZADI, Shahram ; BRIGNULL, Harry ; RODDEN, Tom ; ROGERS, Yvonne ; UNDERWOOD, Mia: Dynamo: A Public Interactive Surface Supporting the Cooperative Sharing and Exchange of Media. In: *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology*. New York, NY, USA : ACM, 2003 (UIST '03), S. 159–168
- [Kammer 2013] KAMMER, Dietrich: *Gesture Formalization for Multi-touch*. 2013. – URL <http://vi-c.de/geformt/index.html>. – Zugriffsdatum: 10.11.2015
- [Kammer u. a. 2010] KAMMER, Dietrich ; WOJDZIAK, Jan ; KECK, Mandy ; GROH, Rainer ; TARANKO, Severin: Towards a Formalization of Multi-touch Gestures. In: *ACM International Conference on Interactive Tabletops and Surfaces*. New York, NY, USA : ACM, 2010 (ITS '10), S. 49–58
- [Kurose und Ross 2014] KUROSE, James F. ; ROSS, Keith W.: *Computernetzwerke : der Top-Down-Ansatz*. Hallbergmoos : Pearson Deutschland, 2014
- [Limbourg u. a. 2004] LIMBOURG, Quentin ; VANDERDONCKT, Jean ; MICHOTTE, Benjamin ; BOUILLON, Laurent ; FLORINS, Murielle ; TREVISAN, Daniela: UsiXML: A User Interface Description Language for Context-Sensitive User Interfaces. In: *Proceedings of the ACM AVI'2004 Workshop "Developing User Interfaces with XML: Advances on User Interface Description Languages"*, ACM, 2004, S. 55–62

- [Lucero u. a. 2011] LUCERO, Andrés ; HOLOPAINEN, Jussi ; JOKELA, Tero: Pass-them-around: Collaborative Use of Mobile Phones for Photo Sharing. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA : ACM, 2011 (CHI '11), S. 1787–1796
- [MathWorks 2014] MATHWORKS: *Capturing Azimuth, Pitch, and Roll Example*. 2014. – URL <http://www.mathworks.com/matlabcentral/fileexchange/40876-android-sensor-support-from-matlab--r2013a--r2013b-/content/sensorgroup/Examples/html/CapturingAzimuthRollPitchExample.html>. – Zugriffsdatum: 25.09.2015
- [Microsoft 2013] MICROSOFT: Cross screen engagement: Multi-screen pathways reveal new opportunities for marketers to reach and engage consumers / Microsoft Advertising, Flamingo & Ipsos OTX. 2013 (3526). – Forschungsbericht
- [Nagel und Fischer 2013] NAGEL, Wolfram ; FISCHER, Valentin: *Multiscreen Experience Design: Prinzipien, Muster und Faktoren für die Strategieentwicklung und Konzeption digitaler Services für verschiedene Endgeräte*. Schwäbisch Gmünd : digiparden GmbH, 2013
- [NFC Forum 2015a] NFC FORUM: *Our Mission & Goals*. 2015. – URL <http://nfc-forum.org/about-us/the-nfc-forum/>. – Zugriffsdatum: 23.09.2015
- [NFC Forum 2015b] NFC FORUM: *What It Does*. 2015. – URL <http://nfc-forum.org/what-is-nfc/what-it-does/>. – Zugriffsdatum: 23.09.2015
- [Nielsen u. a. 2014] NIELSEN, Heidi S. ; OLSEN, Marius P. ; SKOV, Mikael B. ; KJELDSKOV, Jesper: JuxtaPinch: An Application for Collocated Multi-device Photo Sharing. In: *Proceedings of the 16th International Conference on Human-computer Interaction with Mobile Devices & Services*. New York, NY, USA : ACM, 2014 (MobileHCI '14), S. 417–420
- [Oliveira u. a. 2011] OLIVEIRA, João ; GUERREIRO, Tiago ; NICOLAU, Hugo ; JORGE, Joaquim ; GONÇALVES, Daniel: BrailleType: Unleashing Braille over Touch Screen Mobile Phones. In: *Proceedings of the 13th IFIP TC 13 International Conference on Human-computer Interaction - Volume Part I*. Berlin, Heidelberg : Springer-Verlag, 2011 (INTERACT'11), S. 100–107
- [Rawassizadeh u. a. 2014] RAWASSIZADEH, Reza ; PRICE, Blaine A. ; PETRE, Marian: Wearables: Has the Age of Smartwatches Finally Arrived? In: *Commun. ACM* 58 (2014), Dezember, Nr. 1, S. 45–47
- [Rekimoto 1996] REKIMOTO, Jun: Tilting Operations for Small Screen Interfaces. In: *Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology*. New York, NY, USA : ACM, 1996 (UIST '96), S. 167–168

- [Rekimoto 1997] REKIMOTO, Jun: Pick-and-drop: A Direct Manipulation Technique for Multiple Computer Environments. In: *Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology*. New York, NY, USA : ACM, 1997 (UIST '97), S. 31–39
- [Rekimoto 2004] REKIMOTO, Jun: SyncTap: Synchronous User Operation for Spontaneous Network Connection. In: *Personal Ubiquitous Comput.* 8 (2004), Mai, Nr. 2, S. 126–134
- [Roy u. a. 2015] ROY, Nirupam ; GOWDA, Mahanth ; CHOUDHURY, Romit R.: Ripple: Communicating Through Physical Vibration. In: *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*. Berkeley, CA, USA : USENIX Association, 2015 (NSDI'15), S. 265–278
- [Rupp u. a. 2012] RUPP, Chris ; QUEINS, Stefan ; SOPHISTEN, Die: *UML 2 glasklar: Praxiswissen für die UML-Modellierung*. 4. Auflage. München : Hanser Verlag, 2012
- [Seyed u. a. 2012] SEYED, Teddy ; BURNS, Chris ; COSTA SOUSA, Mario ; MAURER, Frank ; TANG, Anthony: Eliciting Usable Gestures for Multi-display Environments. In: *Proceedings of the 2012 ACM International Conference on Interactive Tabletops and Surfaces*. New York, NY, USA : ACM, 2012 (ITS '12), S. 41–50
- [Shirazi u. a. 2009] SHIRAZI, Alireza S. ; WINKLER, Christian ; SCHMIDT, Albrecht: Flashlight Interaction: A Study on Mobile Phone Interaction Techniques with Large Displays. In: *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*. New York, NY, USA : ACM, 2009 (MobileHCI '09), S. 93:1–93:2
- [Terrenghi u. a. 2009] TERRENGHI, Lucia ; QUIGLEY, Aaron ; DIX, Alan: A Taxonomy for and Analysis of Multi-person-display Ecosystems. In: *Personal Ubiquitous Comput.* 13 (2009), November, Nr. 8, S. 583–598
- [Tidwell 2010] TIDWELL, Jenifer.: *Designing interfaces*. Sebastopol, CA : O'Reilly, 2010
- [Walker 2015] WALKER, Geoff: Touch Sensing. In: BHOWMIK, Achintya K. (Hrsg.): *Interactive Displays*. John Wiley & Sons, Ltd, 2015
- [Want 2006] WANT, R.: An introduction to RFID technology. In: *Pervasive Computing, IEEE* 5 (2006), Jan, Nr. 1, S. 25–33
- [van Welie 2008] WELIE, Martijn van: *Patterns in Interaction Design*. 2008. – URL <http://www.welie.com/patterns>. – Zugriffsdatum: 17.09.2015
- [Wi-Fi Alliance 2015a] WI-FI ALLIANCE: *Who We Are*. 2015. – URL <http://www.wi-fi.org/who-we-are>. – Zugriffsdatum: 23.09.2015

- [Wi-Fi Alliance 2015b] WI-FI ALLIANCE: *Wi-Fi Direct*. 2015. – URL <http://www.wi-fi.org/discover-wi-fi/wi-fi-direct>. – Zugriffsdatum: 23.09.2015
- [Wikipedia 2015] WIKIPEDIA: *Bluetooth Low Energy*. 2015. – URL [https://de.wikipedia.org/w/index.php?title=Bluetooth\\_Low\\_Energy&oldid=146099460](https://de.wikipedia.org/w/index.php?title=Bluetooth_Low_Energy&oldid=146099460). – Zugriffsdatum: 06.11.2015
- [Wilson 2005] WILSON, Andrew D.: PlayAnywhere: A Compact Interactive Tabletop Projection-vision System. In: *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology*. New York, NY, USA : ACM, 2005 (UIST '05), S. 83–92
- [Wilson und Sarin 2007] WILSON, Andrew D. ; SARIN, Raman: BlueTable: Connecting Wireless Mobile Devices on Interactive Surfaces Using Vision-based Handshaking. In: *Proceedings of Graphics Interface 2007*. New York, NY, USA : ACM, 2007 (GI '07), S. 119–125
- [Yahoo 2015] YAHOO: *Yahoo Design Pattern Library*. 2015. – URL <https://developer.yahoo.com/ypatterns/>. – Zugriffsdatum: 17.09.2015
- [Yoon 2014] YOON, Sang H.: Designing New Input Modalities for Wearables & Digitized Home. In: *Proceedings of the 2014 ACM International Symposium on Wearable Computers: Adjunct Program*. New York, NY, USA : ACM, 2014 (ISWC '14 Adjunct), S. 151–154

## **Anhang A**

### **Patterntemplate SysPlace**

Beispiel des Pattern-Templates, hier für das Pattern „Bump To Exchange“. Erstellt von Benjamin Grab im Rahmen seiner Bachelorarbeit im Projekt SysPlace.

# Bump To Exchange

## Was

### Problem

Ein Daten-Austausch soll zwischen zwei (oder mehr) Geräten stattfinden.

### Lösung

Ein Quell-Gerät wird mit Ziel-Gerät leicht zusammengestoßen/gebumppt. Dadurch wird ein Datensatz zwischen den Geräten ausgetauscht.

### Grafische Darstellung



### Kategorie

☐ Give | ☐ Take | ☒ Exchange | ☐ Extend | ☐ Connect

## Wie

### Aktion des Benutzers

Zwei Benutzer halten jeweils ein Gerät fest in der Hand und lassen die Geräte an der Seitenkante zusammenstoßen/bumpen.

## **Reaktion des Sende-und Empfänger-Gerätes**

Die Geräte geben nach dem Zusammenstoß eine visuelle oder akustische Rückmeldung an die Benutzer.

Es wird signalisiert, dass ein Bump vom Endgerät festgestellt wurde z.B. über Vibration.

Auf dem Quellgerät angezeigte Dokumente werden auf das Zielgerät übertragen und dort angezeigt.

### **Reaktion im Erfolgsfall:**

- ☒ visuelle Rückmeldung
- ☐ akustische Rückmeldung, z.B. Ton
- ☒ sensitive Rückmeldung, z.B. Vibration

### **Reaktion im Nicht-Erfolgsfall:**

- ☐ wenn keine funktionierende Verbindung besteht ...
- ☐ wenn das Zielgerät nicht erkannt wurde ...
- ☐ wenn das Zielgerät nicht kompatibel ist ...

## **Hinweise zur Gestaltung der Interaktion**

- Es sollte im besten Fall keinen Abstand zwischen den beiden Geräten geben, um die Erkennung des angelegten Gerätes zu gewährleisten.
- Der Zusammenstoß der Geräte muss so erfolgen, dass eine Erkennung möglich ist, andererseits aber die Geräte nicht beschädigt werden.

## **Wann**

### **Geeigneter Nutzungskontext**

...

### **Zeit**

- ☒ gleichzeitige Nutzung von Geräten
- ☐ sequentielle Nutzung von Geräten

### **Modus**

- ☒ online
- ☐ offline

**Ort**

- ☒ privat
- ☒ halb-öffentlich
- ☒ öffentlich
- ☒ stationär
- ☒ unterwegs

**Position**

- ☐ Lean-Back
- ☒ Lean-Forward

**Teilnehmer**

- ☐ Einzelnutzer
- ☒ Kollaboration

**Anordnung zwischen Sender und Empfänger**

- ☒ Face-To-Face
- ☐ Side-To-Side
- ☐ Corner-To-Corner

**Tätigkeit**

- ☒ kleinere Aufgabe
- ☐ wiederholte Tätigkeit
- ☐ ortsbezogene Informationsbeschaffung
- ☐ Ablenkung
- ☐ dringendes

**Abzuratender Nutzungskontext**

— keine Information —

**Geräteklassen**

nach von	Smartwatch	Smartphone	Tablet	Tabletop	Screens
Smartwatch					
Smartphone		x	x	x	
Tablet			x	x	
Tabletop					
Screens					



## Warum

- ☒ Bewährtes Interaction Pattern
- ☐ Interaction Pattern Kandidat: ☐ realisierbar oder ☐ futuristisch

## Analoge Patterns

- Bump To Give
- Bump To Exchange
- Bump To Connect
- Nudge

## State of the Art/Gebrauchshistorie

1. Bump App [1]: Bis 2014 in den App/Play Stores erhältlich gewesen.

## Checkliste: Entspricht die Interaktion der Definition eines "Blended Interaction"?

- ☒ Werden die Designprinzipien berücksichtigt?
  - Die Interaktion findet in Kombination mit physikalischen Gegenständen statt.
  - Die Interaktion kann in einer Kollaboration ausgeführt werden.
  - Die Interaktion unterstützt einen Workflow/eine Aufgabe.
  - Die Interaktion findet in einer physikalischen Umgebung statt.
- ☒ Image Schema/ta liegen zu Grunde.
  - ☒ Container
  - ☐ In-Out
  - ☐ Path
  - ☐ Source-Path-Goal
  - ☐ Up-Down
  - ☒ Left-Right
  - ☐ Near-Far
  - ☐ Part-Whole

- ☒ Die real-weltlichen Kenntnisse des Menschen werden berücksichtigt.
- ☒ Naive Physik
- ☒ Body Awareness and Skills
- ☒ Environment Awareness and Skills
- ☒ Social Awareness and Skills
- ☒ Es ist eine natürliche Interaktion. Metapher/Assoziation: Fistbump, Anstoßen (wie mit Getränken)

## **Technisches**

### **Technologien zur Objekterkennung**

- ☒ 0cm bis 50cm, z.B. NFC, Tags, RFID ...
- ☒ 0,5cm bis 1m, z.B. Bluetooth
- ☐ 1m bis 4m, z.B. Kamera
- ☐ über 4m, z.B. GPS

-

### **Technologien zur Kommunikation**

- ☒ Server-basiert
- ☒ Ad-hoc-Netzwerk basiert

-

### **Technologien zur Bewegungs-/Orientierungsbestimmung**

- ☒ Accelerometer
- ☐ GPS
- ☐ Gyroskop
- ☐ Annäherungssensor
- ☐ Höhenmesser
- ☒ Beacons
- ☐ andere

-

## **Prototyp/Lösungsansatz/Code-Snippets/UML-Diagramm**

Generell können zwei Lösungsansätze unterschieden werden:

- Kommunikation über ein Ad-Hoc Netzwerk, das zwischen den beteiligten Geräten gebildet wird.

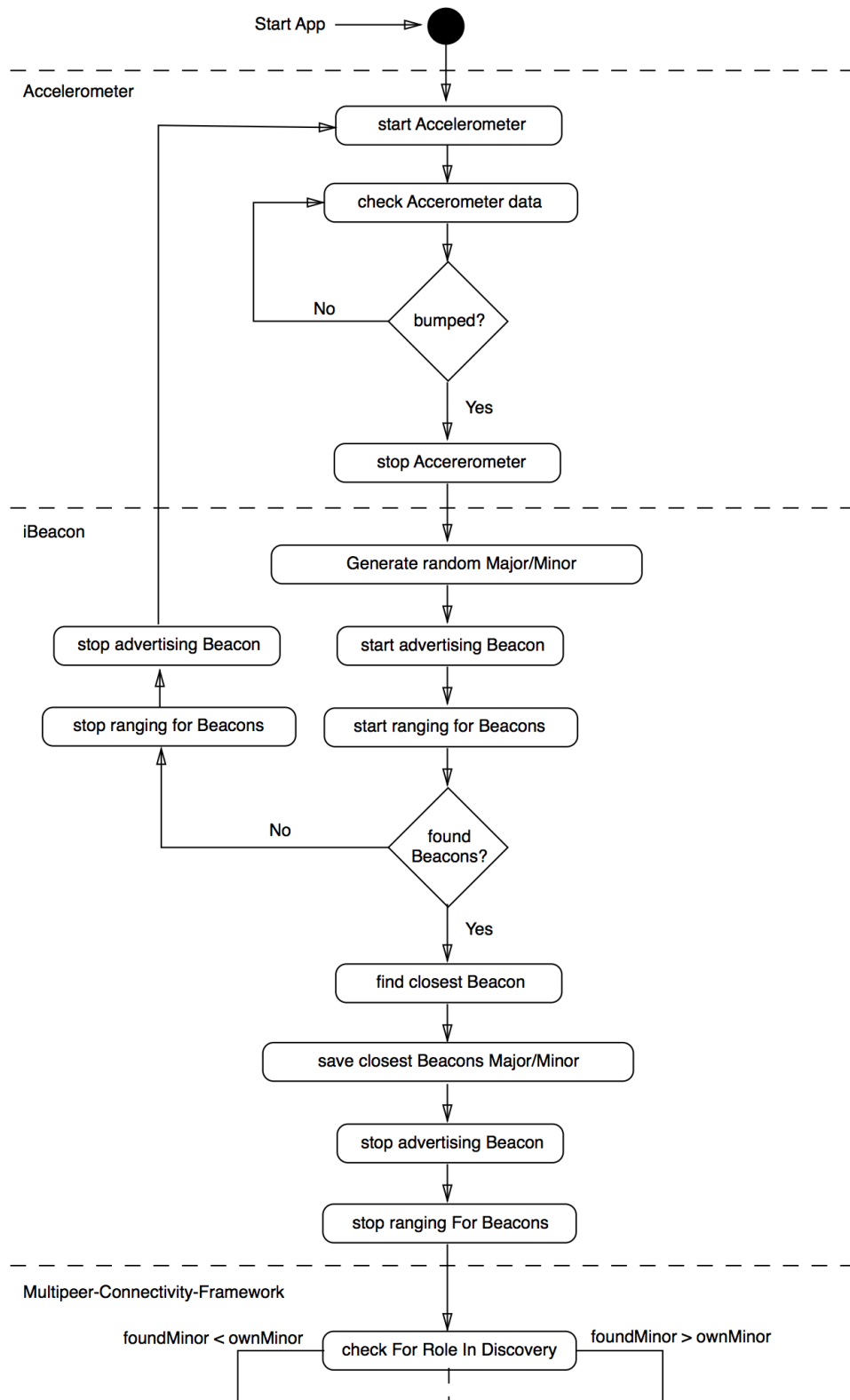
Vorraussetzungen für Endgeräte:

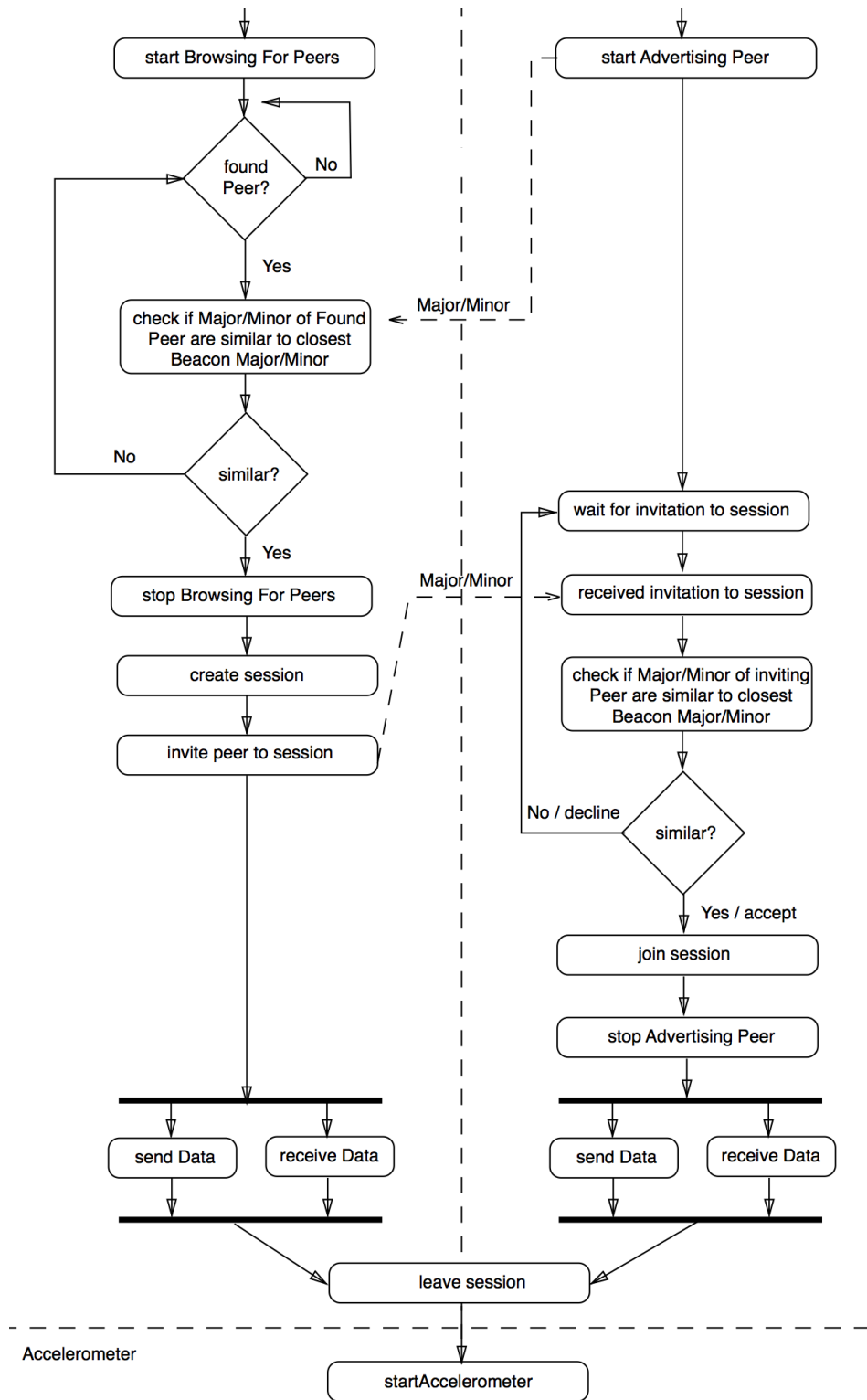
- Accelerometer
- Ad-Hoc-Netzwerktechnologie - Bluetooth/Wi-Fi
- GPS oder Beacon

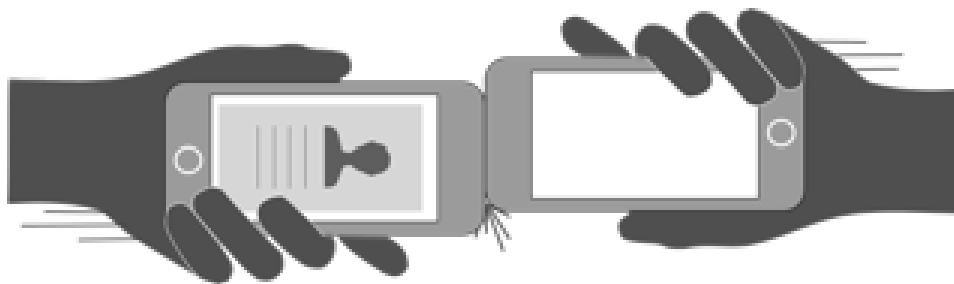
- Kommunikation über einen Server zu dem sich beteiligte Geräte verbinden.

Vorraussetzungen für Endgeräte:

- Accelerometer
- Wi-Fi oder Mobilfunk
- GPS oder Beacon







## **Sonstiges**

### **Autor/en**

Benjamin Grab

### **Literaturreferenzen**

1. Bump. [Online]. <http://bu.mp/>
2. BumpTechnologies. Youtube.com. [Online]. <https://www.youtube.com/user/BumpTechnologies>
3. Ken Hinckley. 2003. Synchronous gestures for multiple persons and computers. In Proceedings of the 16th annual ACM symposium on User interface software and technology (UIST '03). ACM, New York, NY, USA, 149-158. DOI=10.1145/964696.964713

### **Abbildungsverzeichnis**

...

### **Versionshistorie**

Letzte Änderung am: ...

Erstelldatum: ...

### **Kommentare**

...

### **Offene Fragen**

...





## **Anhang B**

# **Übersicht SysPlace-Patterns**


Übersicht über alle SysPlace-Patterns und kurze Beschreibung der jeweiligen Ges-  
ten. Erstellt von Valentina Burjan im Rahmen des Projektes SysPlace.


## SysPlace Pattern Übersicht

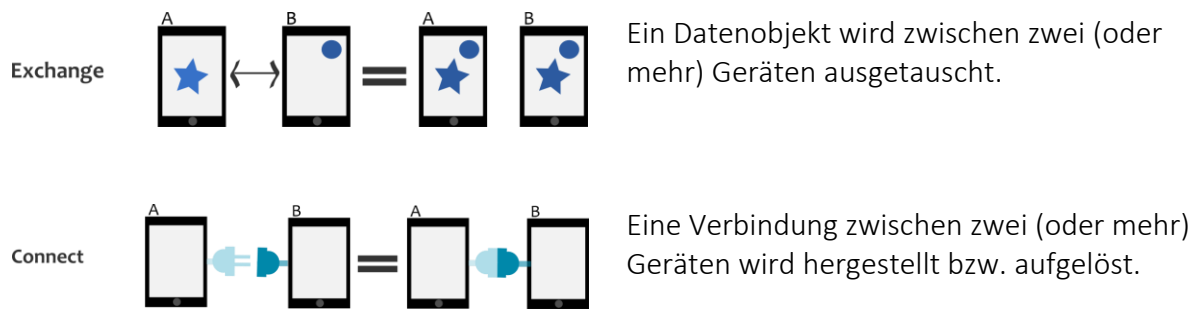
Give	Take	Exchange	Extend	Connect
Give Through Body		Exchange Through Body		
	Grab A Part		Extend With Two Fingers	Shake Well To Connect
Nudge	Grab An Object		Appose	
Stitch To Give	Stitch To Take		Stitch To Extend	Stitch To Connect
Bump To Give	Bump To Take	Bump To Exchange		Bump To Connect
Swipe To Give	Picking Up An Object		World In Miniature	
Dump				
Approach To Give	Approach To Take		Approach To Extend	Approach To Connect
Tennis				Leave To Disconnect
Frisbee				

## Kategorien

**Give**  Ein Datenobjekt wird einem Sender-Gerät auf ein Empfänger-Gerät übertragen.

**Take**  Ein Empfänger-Gerät fordert ein Datenobjekt von einem Sender-Gerät an.  
Das Empfänger-Gerät stellt das Datenobjekt dar.

**Extend**  Ein Datenobjekt auf einem Sender-Gerät wird auf einem (oder mehr) Empfänger-Gerät(en) erweitert/vergrößert dargestellt.



## Patterns

### Appose

Zwei Geräte werden unmittelbar nebeneinander/aneinander gelegt. Hierbei wird die Erweiterung des Displays vom Quell-Gerät auf das Ziel-Gerät veranlasst.

### Approach To Give / Take / Extend / Connect

Ein Nutzer nähert sich mit einem Quell-Gerät einem Ziel-Gerät.

Durch die Nähe wird veranlasst, dass ein Datenobjekt gesendet wird bzw. angefordert wird oder der Bildschirm-Inhalt des Quell-Gerätes auf dem Ziel-Gerät erweitert wird. Außerdem kann durch die Nähe zwischen zwei (oder mehr) Geräten ein Verbindungsaufbau veranlasst werden.

### Bump To Give / Take / Exchange / Connect

Ein Sende-Gerät wird mit einem Empfänger-Gerät leicht zusammengestoßen/gebumpft.

Dadurch wird ein Datenobjekt von einem Sender- zu einem Empfänger-Gerät gesendet bzw. ein Datenobjekt angefordert bzw. ein Datenobjekt ausgetauscht oder eine Verbindung zwischen zwei Geräten hergestellt.

### Dump

Der Anwender hält ein Quell-Gerät fest (ggf. mit beiden Händen) und macht eine Kipp-Bewegung in Richtung des Ziel-Gerätes und überträgt dadurch das Datenobjekt vom Quell-auf das Ziel-Gerät.

### Extend With Two Fingers

Ein Anwender platziert zwei Geräte unmittelbar nebeneinander und berührt mit je einem Finger die Displays der Geräte. Die beiden aufliegenden Finger zieht er zusammen. Dadurch wird eine Erweiterung des Bildschirminhalts des Ziel-Geräts auf das Quell-Gerät veranlasst.

### Frisbee

Der Benutzer hält Sender-Gerät in der Hand und schwenkt es waagerecht aus dem Handgelenk heraus in Richtung des Empfänger-Gerätes. Dabei wird ein Datenobjekt vom Quell-Gerät auf das Ziel-Gerät übertragen.

### Give Through Body / Exchange Through Body

Ein Anwender bildet durch seinen Körper eine Schnittstelle zwischen zwei Geräten, indem er die Hände auf je ein Gerät bzw. dessen Display auflegt und so Daten übertragen bzw. ausgetauscht werden.

### Grab A Part / Grab An Object

Durch Auflegen eines Ziel-Gerätes auf ein Quell-Gerät wird ein ausgewähltes Datenobjekt bzw. ein Teil eines Objektes vom Quell- auf das Ziel-Gerät aufgesaugt. Nach dem Wiederwegnehmen ist das Datenobjekt auch auf dem Ziel-Gerät verfügbar.

### Leave To Disconnect

Der Nutzer veranlasst durch das Verlassen einer definierten Distanz den Verbindungsabbruch zwischen zwei (oder mehr) Geräten.

### Nudge

Ein Sender-Gerät berührt/stupst mit einer seiner Ecken ein Ziel-Gerät auf dessen Displayfläche. Dabei wird die Übertragung eines Datenobjektes vom Quell- auf das Ziel-Gerät ausgelöst.

### Picking Up An Object

Ein Benutzer schiebt/sammelt von einem Quell-Gerät ein Datenobjekt (repräsentiert durch Licht) auf seine Hand und transportiert dieses Lichtobjekt zu dem Ziel-Gerät und legt es ab. Das Datenobjekt ist nun auch auf dem Ziel-Gerät verfügbar.

### Shake Well To Connect

Ein Anwender hat zwei handliche Geräte fest in einer Hand. Durch eine Schüttel-Bewegung wird eine Verbindung zwischen den zwei Geräten hergestellt.

### Stitch To Give / Take / Extend / Connect

Zwei Geräte werden nebeneinandergelegt. Der Anwender zieht mit dem Finger eine Linie auf dem Display des Quell-Gerätes hinüber auf das Display des Ziel-Gerätes, wo die gezogene Linie endet.

Dabei wird ein Datenobjekt vom Sender- auf das Empfänger-Gerät übertragen bzw. ein Datenobjekt angefordert oder der Bildschirm-Inhalt vom Quell- auf der Ziel-Gerät erweitert oder eine Verbindung zwischen den zwei Geräten aufgebaut.

### Swipe To Give

Ein Nutzer hat ein Quellgerät in der Hand oder vor sich stehen. Durch eine Wischbewegung (Swipe) auf dem Display werden Daten auf ein zweites Gerät, das Zielgerät, übertragen. Die Übertragung und die Swipe-Richtung hängen mit der Ausrichtung der Geräte zusammen.

### Tennis

Ein Benutzer hält ein Sender-Gerät fest in der Hand und schwenkt es aus dem Unterarm heraus in Richtung des Empfänger-Gerätes. Hierbei wird ein Datenobjekt vom Quell-Gerät auf das Ziel-Gerät übertragen.

### World In Miniature

Ein Anwender hat ein Ziel-Gerät in der Hand. Das Ziel-Gerät ermöglicht die erweiterte Ansicht (detailliert bzw. grob) des Bildschirm-Inhaltes eines Quell-Gerätes.



## **Anhang C**

# **Abwandlungen der synchronen Gestenerkennung**

Zwei Alternativen für die synchrone Gestenerkennung:

- Abbildung C.1: Erkennung einer synchronen Geste mit einem Gerät als Erkennungsserver.
- Abbildung C.2: Parallele Gestenerkennung auf jedem Gerät.

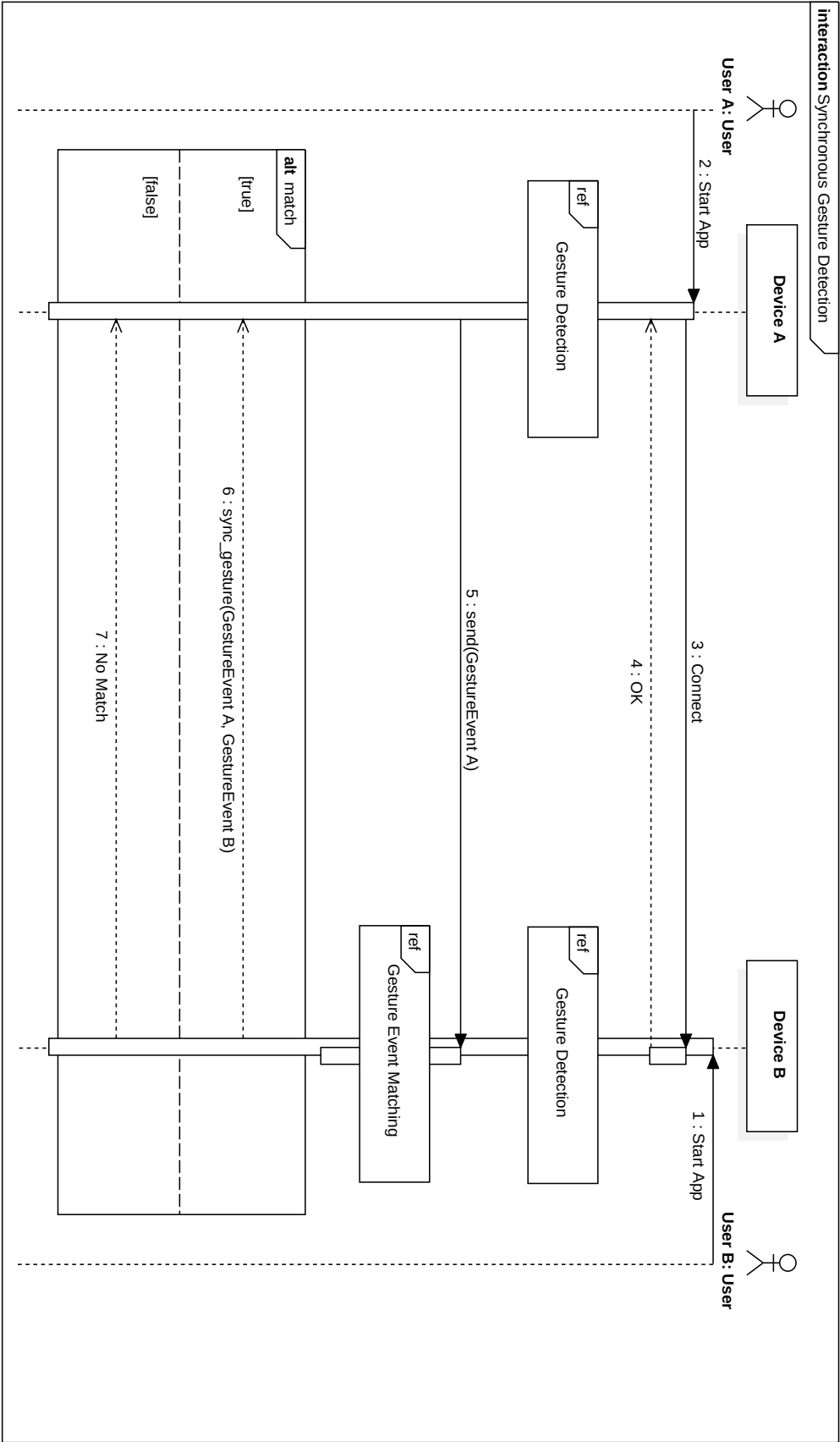
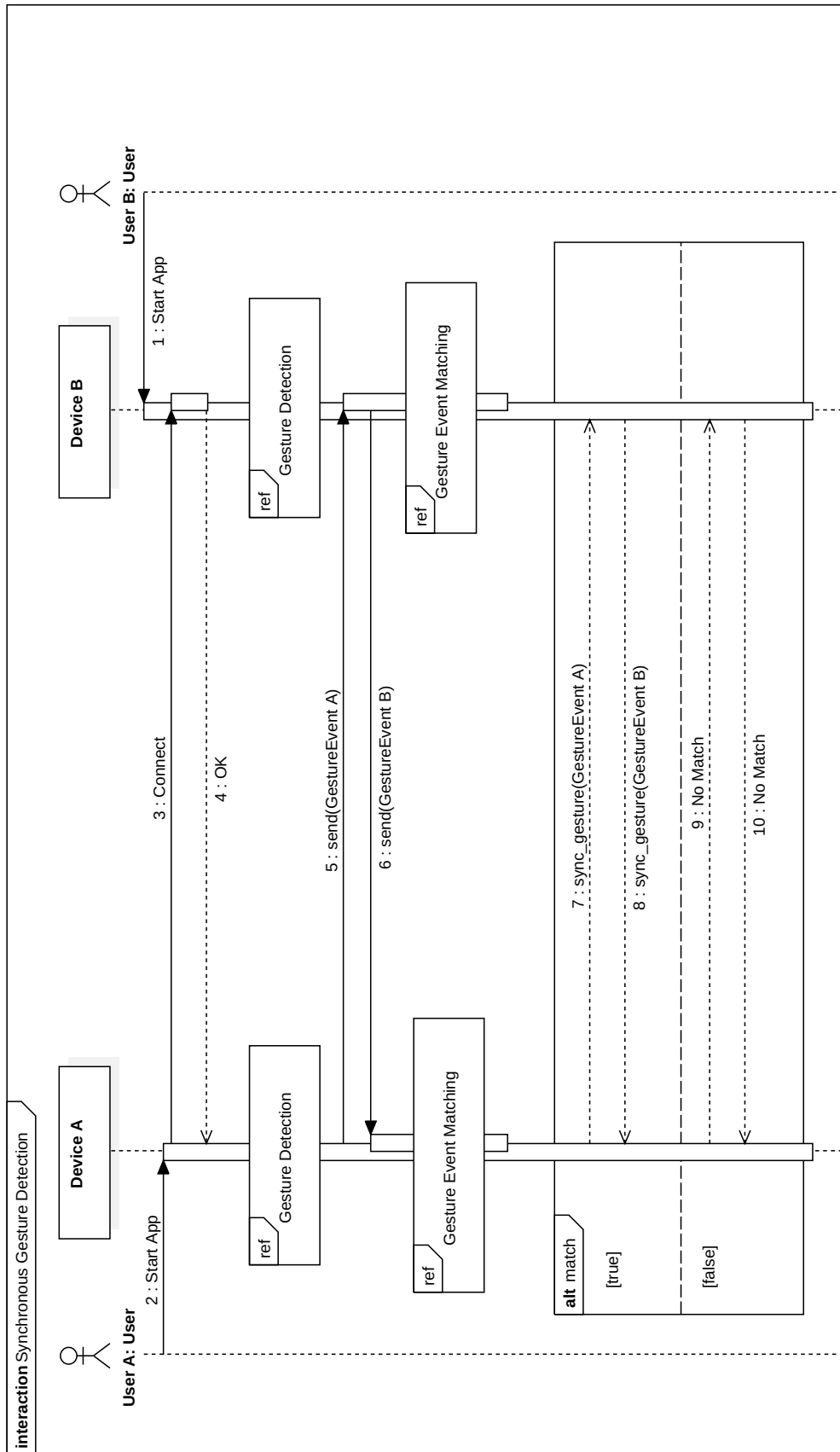


Abbildung C.1.: Erkennung einer synchronen Geste mit einem Gerät als Erkennungsserver.





**Abbildung C.2.:** Parallele Gestenerkennung auf jedem Gerät.