Fachhochschule Nordwestschweiz
Hochschule für Technik

Institut für Mobile und Verteilte Systeme

FOKUS REPORT

2015

9 771662 201005 55

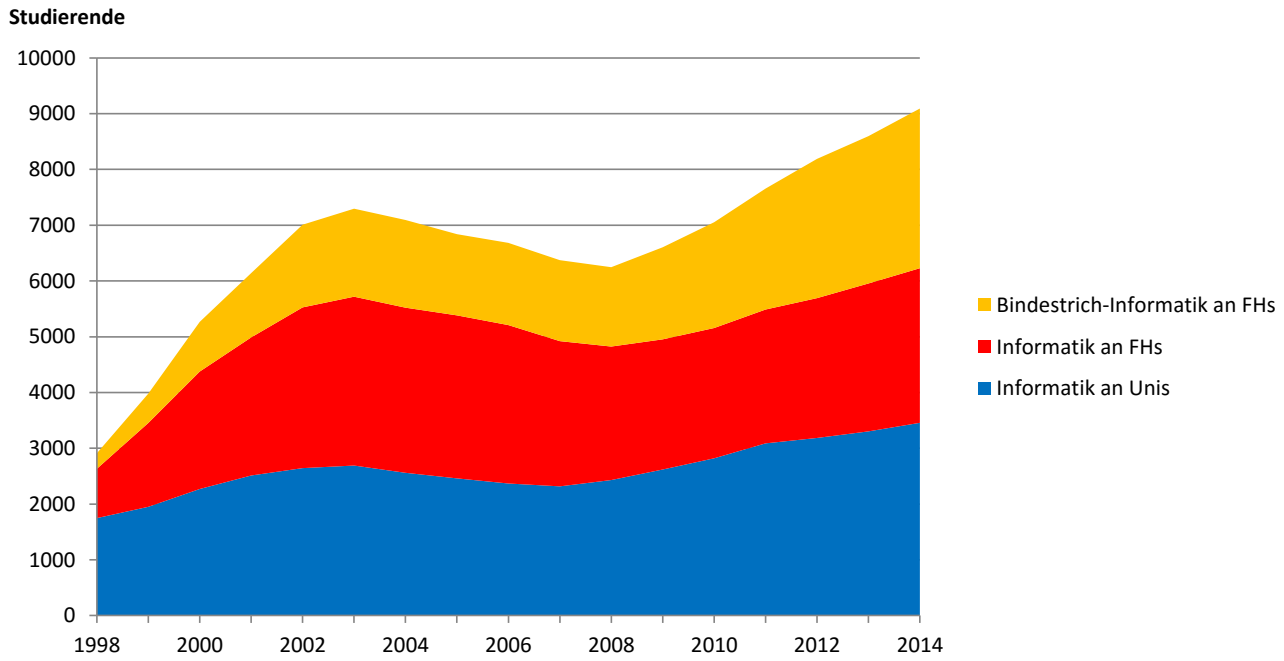

Abbildung 1: Informatikstudierende der Schweizer universitären Hochschulen und Fachhochschulen. Nach einem Rückgang der Studierendenzahlen in den Jahren 2003 bis 2008 ist das Interesse an Informatik wieder steigend. Während die klassische Informatik an den FHs fast wieder bei den Werten von 2002 angekommen ist, hat sich die Bindestrich-Informatik gegenüber dem Jahr 2002 quasi verdoppelt. (Quelle: Bundesamt für Statistik)

Bindestrich

Eine der grossen Stärken der technischen Studiengänge der Fachhochschulen in der Schweiz ist deren Nähe zu Industrie und Wirtschaft. Absolventinnen und Absolventen finden sich schnell zurecht und ihre Arbeitgeber können mit einem raschen produktiven Einsatz rechnen. In Projektteams treffen sie oft auf Kollegen mit ähnlicher Bildungslaufbahn, aber manchmal auch auf neue Kolleginnen, die komplementäres Wissen ins Team miteinbringen. Anhand der eingebrachten Qualitäten und Kernkompetenzen lernen sich die Teammitglieder gegenseitig wertschätzen. Diese Wertschätzung muss jedoch schrittweise erarbeitet werden, denn nicht selten betrachten sich Teammitglieder anfänglich unter dem Aspekt von Vorurteilen über Studiengänge oder auch ganzer Hochschulen. Die komplementäre Zusammensetzung von Projektteams ist also alles andere als problemlos und sorgt in solchen Teams immer wieder mal für Unverständnis. Daher erstaunt es nicht, dass es Informatikfirmen gibt, die ihre Belegschaft ziemlich homogen rekrutieren.

Nicht nur in den letztgenannten Firmen, sondern ganz allgemein stellt sich in Software-Projektteams immer wieder die Frage, wer und wie die Anwendungsdomäne in einem Projekt vertreten soll. Während hauptsächlich im Projektgeschäft tätige Firmen überzeugt sind, dass sie mit strukturierten Vorgehensmodellen und unter dem regelmässigen Einbezug der Anwender deren Kompetenz extern ins Projektteam integrieren können, gibt es andere Firmen, die der Meinung sind, dass ein möglichst grosser Teil ihrer Entwickler die Anwendungskompetenz selber aufweisen und am liebsten gleich schon nach dem Studium mitbringen soll.

Ausgehend von der These – die in letzter Zeit wieder viel öfter vertreten wird –, dass Software-Entwickler schon möglichst viel Wissen aus einer Anwendungsdomäne mitbringen sollen, um sich schneller in ein entsprechendes Projekt oder Team einbringen zu können, stellt sich nun die Frage, ob Hochschulen anstatt „reiner" Informatiker nicht besser sogenannte Bindestrich-Informatikerinnen ausbilden, also den Studierenden nicht nur die Informatik beibringen, sondern ihnen auch gleich noch ein Anwendungsdomänenwissen ausserhalb der Informatik mitgeben sollten. Neben bekannten und etablierten Disziplinen wie Wirtschafts-Informatik sind in den letzten Jahren weitere Informatikdisziplinen entstanden und es ist damit zu rechnen, dass noch weitere entstehen werden (siehe Abb. 1, Umschlaginnenseite). Es ist naheliegend, dass sich beispielsweise für ein Studium der Biomedizin-Informatik zusätzliche junge Leute begeistern liessen, welche sonst für

Inhalt

Impressum

Herausgeberin:
Fachhochschule Nordwestschweiz FHNW
Institut für Mobile und Verteilte Systeme
Bahnhofstrasse 6
CH-5210 Brugg-Windisch
www.fhnw.ch/technik/imvs
Tel +41 56 202 99 33

Kontakt: Prof. Dr. Jürg Luthiger
juerg.luthiger@fhnw.ch
Tel +41 56 202 78 23
Fax +41 56 462 44 15

Redaktion: Prof. Dr. Christoph Stamm
Layout: Claude Rubattel
Erscheinungsweise: jährlich
Druck: jobfactory Basel
Auflage: 150

ISSN 1662-2014 (Print)
ISSN 2296-4169 (Online)

das oft als trocken empfundene Informatikstudium nicht gewonnen werden könnten. Klingt dieser Ansatz mit dem komplementären Domänenwissen auf Anhieb plausibel und verlockend, so sind doch ein paar Fragezeichen angebracht: Welche Aufgaben sollen komplementär Ausgebildete in einem Projektteam oder einer Firma primär übernehmen? Verfügen sie über genügend Software-Kompetenz, um gute Programme im entsprechenden Anwendungsbereich zu entwickeln oder werden sie mehr eine Vermittlerrolle zwischen Anwenderseite und Informatikseite übernehmen? Für welche Anwendungsdomänen sollen generell Bindestrich-Informatikerinnen ausgebildet werden? Und weshalb keine für die anderen? Bleiben diese Anwendungsbereiche über eine längere Zeit konstant?

Offensichtlich haben in den letzten Jahrzehnten immer mehr Anwendungsbereiche von den grossartigen Leistungen der Informatik profitieren können und somit ist es kaum verwunderlich, dass heute beinahe jedermann ein gewisses Mass an Informatikwissen benötigt, um in der modernen Geschäftswelt teilnehmen und die gröbsten Zusammenhänge verstehen zu können. So wie ein Grundstock an Mathematik erlaubt, abstrakt zu denken und funktionale Abhängigkeiten zu formulieren, so ist auch ein Verständnis für automatisierte Informationsverarbeitung in unserer computerisierten Welt sehr nützlich und hilfreich.

Ist der Ruf nach Bindestrich-Informatikern nun einfach eine direkte Folge der fehlenden Informatikgrundkenntnisse in weiten Bevölkerungsschichten, welcher wieder verschwinden wird, sobald alle jungen Menschen mit einem Rucksack an Informatik ausgestattet sind, oder gibt es einen echten und ausgewiesenen Bedarf dafür, der über mehrere Jahrzehnte absehbar ist? Ist damit zu rechnen, dass ständig neue Bindestrich-Studiengänge für eine kurze Zeit zum Leben erwachen und beim ersten Rückgang an Studierenden durch eine andere Spezialität ersetzt werden? Liesse sich damit (noch) mehr Praxisnähe im Studium erzielen und mehr Flexibilität der Hochschulen einfordern? Weshalb nicht gleich der ganz grosse Wurf: das vollständig individualisierte Studium? Jede lernwillige Person kriegt ein auf ihre persönlichen Bedürfnisse angepasstes Rundumpaket, bestehend aus Coaching, Online-Learning, Socializing-Events und Teamworkshops. Bei genauer Betrachtungsweise sind wir davon nicht mehr weit entfernt. Bereits heute können die Studierenden ihr Studium aus einer grossen Fülle an verschiedenen Modulen individuell zusammenstellen und der Übergang von gemeinsamen Lehrveranstaltungen zu Online-Kursen hat gerade eben begonnen. Bis zum individualisierten Studienabschluss fehlt wirklich nicht mehr viel.

Sind die grossen Auswirkungen der Informatik auf weite Anwendungsbereiche nicht Beweis genug, dass fundiert ausgebildete Technikerinnen und Informatiker die Gesellschaft und das Leben nachhaltig verändern können? Oder werden diese Veränderungen nur negativ wahrgenommen? Weshalb soll von diesem Erfolgsmodell abgewichen werden? Ist etwa das Ende der Computertechnik bereits erreicht und geht es jetzt lediglich noch darum, diese Technik in weiten Kreisen zur Anwendung zu bringen? Vielleicht ist es aber auch gerade das Gegenteil davon, nämlich das frühzeitige Besetzen von ergiebigen Entwicklungsfeldern. Letztlich steht aber die grosse Frage im Raum: Wer darf bzw. soll was studieren? Ohne den Anspruch zu erheben, diese grundlegende Frage hier beantworten zu wollen oder zu können, ist seitens der Industrie und Wirtschaft der offensichtliche Wunsch vorhanden, dass junge Menschen sich vermehrt an Mathematik, Informatik, Ingenieursdisziplinen und Naturwissenschaften (MINT) heranwagen. Es geht also darum, bei Jugendlichen, die vor der Berufs- oder Studienwahl stehen, vermehrt eine Interessensverlagerung hin zu den MINT-Disziplinen zu bewirken. Dazu bedarf es sowohl interessanter Berufsbilder an der Schnittstelle zwischen Mensch und Maschine als auch guter und attraktiver Angebote, die eine klare berufliche Perspektive aufweisen.

Bindestrich-Informatik weist zwar auf einen interessanten und gangbaren Weg hin, um die angesprochene Verlagerung zu bewirken, jedoch meine ich, dass es den Bindestrich nicht wirklich braucht! Natürlich erfordern viele Berufe und Studiengängen fundiertes Informatikwissen, so wie es auch Mathematik braucht. Das alleine ist aber noch lange kein Grund dafür, überall das Etikett Informatik dranzuhängen und damit den Begriff Informatik noch weiter zu verwässern. Stattdessen sollten neue und einprägsame Berufsbilder entwickelt werden, die selbstverständlich auf ein breites Grundlagenangebot von verschiedenen Informatikteildisziplinen zugreifen dürfen. Aber eine vermehrte Einführung von Bindestrich-Studiengängen, welche weder die Informatik noch die Anwendungsdomäne zweckmässig auszubilden vermögen, wird mit den Nachteilen einer Oberflächlichkeit bezahlt werden müssen, die heute schon überall dort ersichtlich ist, wo der Schein dem Sein den Rang abgelaufen hat. Für die Herausforderungen, die uns die demografische Entwicklung stellen wird, brauchen wir nicht primär Leute, die zwischen Informatik und Anwendung vermitteln können, sondern wir brauchen bestens ausgebildete Informatikerinnen, die in der Lage sind, die Automation weiter zu treiben und somit die Produktion von Gütern aber auch Dienstleistungen aufrecht zu erhalten, auch dann, wenn immer weniger junge Menschen einer älter werdenden Gesellschaft gegenüberstehen werden.

Prof. Dr. Christoph Stamm

Neue Forschungsstrategie des IMVS

Das IMVS ist seit 2010 in den drei Forschungsfeldern „Effiziente Software Entwicklung“, „Effiziente und Parallele Software“ und „ICT System- und Service-Management“ aktiv. In diesem Jahr haben wir unsere Forschungsfelder kritisch überprüft und sind zur Überzeugung gekommen, eine Neuausrichtung zu wagen. Die neuen Forschungsfelder und die Gründe, die uns zu dieser Neuausrichtung motivieren, sind nachfolgend beschrieben.

Jürg Luthiger | juerg.luthiger@fhnw.ch

In vergangener Zeit waren unsere Forschungsprojekte in grossem Masse geprägt von der Suche nach Lösungen, um die physikalische Welt und die digitale Welt einander näher zu bringen. Es entstanden Informatik-Systeme mit verteilten Architekturen und mit diversen Applikationen auf unterschiedlichen Endgeräten, wobei Lösungen für Smartphones dominierten.

Die Verbindung der physikalischen Welt mit digitalen Services ist heute unter dem Schlagwort *Internet of Things* (IoT) zu einem Hype angewachsen. Mit IoT wird die Verknüpfung eindeutig identifizierbarer physischer Objekte mit einer virtuellen Repräsentation in einer Internet ähnlichen Struktur bezeichnet. Dabei spielen die Internetprotokolle eine zentrale Rolle, da sie mit ihren bewährten Standards zu einem offenen System führen und ein Ökosystem fördern, das eine hohe Skalierbarkeit und Erweiterbarkeit unterstützt; ein zentrales Anliegen jedes Informatikingenieurs.

Durch die Einbindung der physikalischen Welt wächst die digital verfüg- und nutzbare Menge an Rohdaten enorm an. Meistens verschwinden diese Rohdaten in Datensilos und Datenbanken. Unter dem Begriff *Big Data* werden Technologien zusammengefasst, die aus den gespeicherten Daten durch Verknüpfung, Aggregation, Einbezug des Kontexts usw. nutzbringende Informationen zu generieren versuchen. Es wird also sprichwörtlich die Nadel im Heuhaufen gesucht. Manchmal ist es sinnvoller das frische Gras direkt zu verfüttern, anstatt es zu trocknen und in Silos zu speichern. Das soll heissen, die Rohdaten müssen in Echtzeit angereichert, verknüpft oder konsolidiert werden, damit darauf unmittelbar reagiert werden kann, beispielsweise um physische Prozesse zu steuern und zu optimieren. Systeme und Methoden zu erforschen, die diese Aufgabe unterstützen, erachten wir als eine sehr spannende Herausforderung der Informatik.

IoT-Systeme sind verteilte Systeme, die miteinander kommunizieren sollen. Da die IoT-Welt neue IT-Architekturen hervorbringen wird, die in erster Linie asynchron und nachrichten-basiert sind, wird sich auch der Informatiker von den traditionellen Client-Server-Systemen lösen müssen. Das Internet zeichnet sich durch eine hohe Robustheit aus, die nun auch in die Welt der kleinen Systeme diffundieren soll. Microservices, Reactive Systems und Resilient Systems sind Ansätze, um diesem Anspruch gerecht werden zu können.

Über IoT-Anwendungen werden persönliche Daten, Daten zum Nutzungsverhalten, aber auch Geschäftsgeheimnisse bis hin zu Wirtschaftsdaten abgreifbar. Anreize Daten zu stehlen, unberechtigt zu nutzen, zu manipulieren oder gar zu zerstören gibt es deshalb für zahlreiche Akteure. Dabei ist der Angreifer nicht mehr nur der unbekannte Hacker, sondern oftmals der rechtmässige Benutzer einer IoT-Anwendung. Chiptuning in der Automobilwelt oder Manipulation digitaler Stromzähler in der Energiewirtschaft sind bekannte Beispiele eines solchen Fehlverhaltens. Der Datenschutz in IoT-Anwendungen wird vor allem durch übermässiges Erheben, Verarbeiten und verteiltes Speichern von Daten, unzureichende oder unsichere Zugriffskontrollen, mangelnde Verschlüsselung und schlechte Anonymisierung gefährdet. Die Notwendigkeit eines effektiven Datenschutzes ist deshalb eine grosse und dringende Herausforderung.

Das IMVS wird sich diesen Herausforderungen stellen, da unsere Industrie- und Wirtschaftspartner entsprechende Lösungsansätze erwarten und wir in diesen Themen anspruchsvolle und interessante Forschungsaktivitäten identifizieren können. Mit unseren neuen Forschungsfeldern signalisieren wir unseren Partnern, dass wir uns ständig erneuern und für die Zukunft gerüstet sind, um ihnen bei der Umsetzung ihrer IoT-Projekte ein wertvoller Forschungspartner sein zu können.

Microservice Architektur: Ein Fundament für mobile Applikationen?

Der renommierte Software-Architekt Martin Fowler hat 2014 den Begriff „Microservice Architektur“ geprägt [4]. Mit einem solchen Architektur-Ansatz wird eine komplexe Anwendungssoftware nicht nur in Software-Komponenten aufgeteilt, sondern diese bleiben auch im Betrieb unabhängige Komponenten, die in einem eigenen Prozess laufen und die untereinander lediglich über technologie- und sprachunabhängige Schnittstellen kommunizieren. Die lose Koppelung der Komponenten bietet diverse Vorteile. Deshalb haben in der Zwischenzeit verschiedene grosse Firmen wie Amazon oder Netflix diesen Architekturansatz aufgegriffen. Auch wir haben uns bei der Überarbeitung des hochschuleigenen Informationssystems „App4Technik“ entschlossen, die Gesamtapplikation aus einer monolithischen Software-Architektur in eine Microservice Architektur zu überführen, die sowohl die Server-Applikation wie auch die mobile Technik-App umfasst. In diesem Artikel beschreiben wir unsere Erfahrungen aus dieser Migration.

Jürg Luthiger | juerg.luthiger@fhnw.ch

„App4Technik“ ist ein verteiltes Software-System bestehend aus:

- verschiedenen Modulen bzw. Server-Services wie Menüplan der Mensa, Neuigkeiten aus der Hochschule für Technik, zukünftige Events der Hochschule oder Beschreibung der Studiengänge;
- einem Content Management System (CMS) für die Marketing-Abteilung zur Pflege der News, der Events und den Informationen zu den Studiengängen;
- einer Android- und einer iPhone-App für die Studierenden und die Mitarbeitenden der Hochschule.

Der Server basiert auf einer monolithischen Software-Architektur, d.h. alle Services laufen im gleichen Prozess. Die Services sind in Java implementiert und sie greifen auf die gleiche Datenbank zu. Sie stellen für das CMS und die mobilen Clients gemeinsam je eine entsprechende Programmierschnittstelle (API) zur Verfügung, die über das HTTP-Protokoll genutzt werden kann (Abb.1). Das API orientiert sich am REST-Programmierparadigma [3].

Die bisherigen mobilen Applikationen sind hybride Apps, d.h. dass sie auf einer gemeinsamen JavaScript-Codebasis basieren und mit der PhoneGap-Technologie als Android- oder iOS-App ausgeliefert werden können. Das ganze Informationssystem ist seit 2013 in Betrieb. Die mobilen Applikationen stehen über die beiden App Stores „Google Play“ und „App Store“ zur Installation bereit.

Der produktive Betrieb des Systems hat Verbesserungspotential aufgezeigt, vor allem in Bezug auf eine einfache Wart- und Erweiterbarkeit. Ebenfalls führt jede Anpassung bei den mobilen Applikationen zu einem erneuten Einreichen und der damit verbundenen Überprüfung der Apps in den entsprechenden App-Stores. Da dieser Genehmigungsprozess vor allem bei Apple aufwändig ist, werden Erweiterungen eher zögerlich vorgenommen.

Wir haben uns deshalb entschlossen, die bestehende Applikation im Rahmen einer Bachelor-Arbeit durch die beiden Informatik-Studenten Damian Keller und Matthias Giger zu überarbeiten. Das Ziel der Arbeit ist wie folgt festgelegt worden: „Die bestehende Applikation mit dem monolithischen Ansatz soll in eine Microservice Architektur überführt werden, um den Betrieb und die Erweiterung der einzelnen Dienste zu erleichtern. Dabei ist ein Konzept für die gesamte Applikation zu entwickeln, das eine einfache Integration neuer Dienste und einen einfachen Release-Wechsel bestehender Dienste erlaubt.“ Am Einsatz von PhoneGap soll festgehalten werden. Daher werden folgende Technologien für die neue mobile Applikation vorgegeben:

- *PhoneGap* [6]: PhoneGap ist ein Framework, um mit Web-Technologien wie HTML, CSS und JavaScript mobile Applikationen erstellen zu können. Der grosse Vorteil von PhoneGap ist die Möglichkeit auf Basis eines gemeinsamen Quellcodes mobile Applikationen für unterschiedliche Plattformen generieren zu lassen.
- *Ionic* [5]: Ein Framework für hybride mobile Applikationen mit einem starken Fokus auf das User Interface und die User Interaktion. Ionic Applikationen sollen wie native Applikationen erscheinen. Das Framework arbeitet nahtlos mit PhoneGap und AngularJS zusammen.
- *AngularJS* [1]: Ein modernes, ausgereiftes JavaScript Framework von Google für Rich Internet Applications. Das „App4Technik“-CMS ist bereits mit AngularJS programmiert worden.

Mit der Migration auf die Microservice Architektur versprechen wir uns eine stark verbesserte Flexibilität im Umgang mit bestehenden und neuen Services. Vor allem dann, wenn es uns gelingt diese Flexibilität bis in die mobilen Applikationen hinaus zu führen.

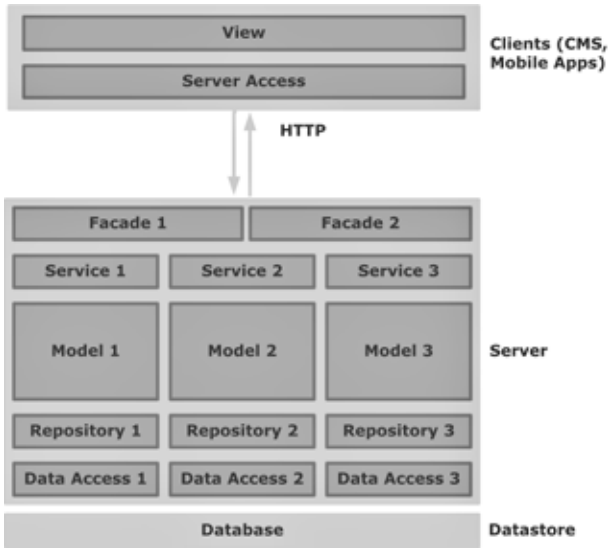

Abbildung 1: Gesamtsystem mit Server als Monolith, separater Datenbank und den Clients (CMS, mobile Applikationen)

Was sind Microservices?

Auf Wikipedia wird der Begriff Microservices wie folgt beschrieben: „Microservices sind ein Architekturmuster der Informationstechnik, bei dem komplexe Anwendungssoftware aus kleinen, unabhängigen Prozessen komponiert werden, die untereinander mit sprachunabhängigen Programmierschnittstellen kommunizieren. Die Dienste sind klein, weitgehend entkoppelt und erledigen eine kleine Aufgabe. So ermöglichen sie einen modularen Aufbau von Anwendungssoftware."

Dank der starken Entkoppelung zeichnen sich Microservices durch folgende Eigenschaften aus:

- Ein Microservice setzt genau einen konkreten Anwendungsfall um.
- Ein Microservice kann einen eigenen Technologie-Stack oder Datenbank nutzen.
- Ein Microservice kann in einer beliebigen Programmiersprache implementiert werden.
- Ein Microservice kann unabhängig von anderen Microservices produktiv betrieben werden.
- Ein Microservice kann einfach ersetzt werden, ohne dass die Gesamtapplikation ausser Betrieb genommen werden muss.

Im Vergleich dazu führt eine enge Koppelung bei einer monolithischen Software-Architektur zu Server-Services mit folgenden Bedingungen:

- Ein Ersatz oder Update eines Server-Services führt zu einer kurzfristigen Nichtverfügbarkeit der Gesamtapplikation, da die Applikation heruntergefahren werden muss, um die neue Funktionalität in Betrieb nehmen zu können;
- Alle Server-Services müssen mit der gleichen Programmiersprache implementiert werden, da sie im gleichen Prozess laufen;
- Alle Server-Services müssen den gleichen Technologie-Stack nutzen.

Die Unabhängigkeit der Microservices untereinander führt zu einer hohen Flexibilität, welche die Microservice-Architektur sowohl für Software-Entwickler wie auch für Software-Betreiber interessant macht.

Konzept für eine flexible „App4Technik"

Um die Microservices bei „App4Technik" einführen zu können, muss zuerst ein neues Architektur-Konzept entwickelt werden, das auf den bestehenden Server-Services und auf den Ideen der Microservice-Architektur basiert.

Wie in Abb. 1 ersichtlich, kann auch mit einer monolithischen Software-Architektur eine saubere Trennung der Anwendungsfälle implementiert werden. Denn jeder Server-Service setzt einen konkreten Anwendungsfall wie Mensa-Service oder News-Service um. Die Server-Services basieren dabei auf einer gängigen Schichten-Architektur mit Service-Layer, den Entitäten als Model und Repository- bzw. Data Access-Layer für den Zugriff auf die Datenbank. Es ist offensichtlich, dass diese Server-Services sehr gute Kandidaten für jeweils einen Microservice darstellen.

Erweitert man nun jeden Server-Service mit einem eigenen REST-API sowie einer eigenen Datenbank und betreibt man diese Server-Services in unabhängigen Prozessen, so ist ein erster Schritt der Entkoppelung vollzogen. Über das REST-API können beliebige Client-Applikationen die Server-Services nutzen. In diesem Projekt sind es zwei: das CMS und die mobile Applikation.

Das CMS ist eine Rich Internet Application (RIA), die für die Marketing Abteilung eine moderne, interaktive Benutzeroberfläche bereitstellt. Im Betrieb hat sich diese Applikation als sehr robust und problemlos wart- und bedienbar erwiesen. Sie wird deshalb nicht auf eine Microservice-Architektur umgeschrieben, sondern als eigenständige Applikation belassen, die über HTTP und über die entsprechenden APIs den Inhalt der Microservices pflegt.

Die mobile Applikation hingegen soll in die Struktur der Microservice-Architektur überführt werden. Es ist wichtig, dass die visuelle Repräsentation und die Funktionalität eines Microservices auf der mobilen Seite über den Microservice selber festgelegt werden kann, so dass der Server und der mobile Teil eine logische Einheit bilden. Zum Beispiel soll ein Microservice nach einer Deaktivierung nicht mehr in der mobilen Applikation erscheinen. Die visuelle Client-Komponente des Microservice hat demnach eine enge Koppelung zum Microservice auf dem Server, auch wenn die beiden Komponenten physisch auf anderen Devices und in unterschiedlichen Prozessen laufen. Logisch gehören sie zusammen.

Zusätzlich müssen die Microservices eine Möglichkeit erhalten, auch untereinander Informationen austauschen zu können. In der Tabelle 1 sind die möglichen Kommunikationsarten aufgelistet.

Aus der Tabelle 1 ist ersichtlich, dass eine one-to-many Kommunikation nur mit einem asynchronen Ansatz realisierbar ist. Eine asynchrone Kommunikation erfordert den Einsatz einer aufwändigen Messaging Infrastruktur, während eine synchrone one-to-one Kommunikation auf Basis des Request/Response Modells von HTTP einfach realisierbar ist.

Um die Kommunikationsart festlegen zu können, ist die bestehenden Applikation analysiert worden. Die Analyse hat gezeigt, dass insgesamt 9 Microservices notwendig sind und dass diese untereinander teilweise abhängig sind. Die Tabelle 2 fasst die notwendigen Microservices zusammen und zeigt auch deren Beziehungen zueinander auf.

Die Tabelle 2 zeigt, dass die Koppelung zwischen den Microservices gering ist. Auch der Informationsaustausch zwischen den Microservices ist einfach und hat eindeutige Kommunikationspartner. Deshalb genügt das einfache, synchrone Request/Response-Kommunikationsmodell von HTTP.

Aus diesen Erkenntnissen kann ein Konzept für den Microservice abgeleitet werden. In Abbildung 2 ist dieses Konzept und die interne Struktur unseres Microservices mit den drei grösseren Software-Komponenten *Service Consumer*, *Service Server* und *Service Datastore* dargestellt. Der *Service Consumer* ist die mobile Komponente und kommuniziert mit seinem *Service Server* über HTTP. Dieser wird mit einer Schichten-Architektur weiter strukturiert. Eine eigene Datenbank ist für die persistente Datenhaltung der Model-Entitäten zuständig. Für die Kommunikation mit anderen Microservices wird ein HTTP-Client als Gateway in die Server-Komponente integriert.

	one-to-one	one-to-man
synchron	Request/Response	—
asynchron	Point-to-Point	Publish/Subscribe

Tabelle 1: Typische Kommunikationsarten

Die Gesamtfunktionalität von „App4Technik“ wird über verschiedene Microservices realisiert, die auf der Server-Seite einzeln betrieben werden können. Auf der mobilen Seite jedoch, müssen die verschiedenen *Service Consumer* zu einer Einheit zusammenfasst werden, so dass der User die Applikation „App4Technik“ über eine einzige App nutzen kann. Es ist deshalb eine weitere Software-Komponente notwendig, die über ein Framework eine beliebige Anzahl von *Service Consumer* aufnehmen kann (siehe Abb. 3).

Microservices auf dem Server

Die Microservices für „App4Technik“ werden auf dem Server als *Spring Boot* Applikationen implementiert. *Spring Boot* ist ein Framework für die einfache Entwicklung eigenständig lauffähiger Spring-Anwendungen, die per „Convention over Configuration», d.h. ohne explizite Konfiguration auskommen und die alle notwendigen Klassenbibliotheken mitbringen, so dass die eigentliche Implementierung erheblich reduziert werden kann.

Die implementierten *Service Servers* nutzen die Unterstützung von *Spring Boot* in grossem Masse. Deshalb wird in diesem Kapitel die Umsetzung der Anwendungsfälle nicht vertieft aufgezeigt. In-

	Microservice	Beschreibung	Nutzt
1	Mensa	Dieser Service stellt der App die Menus der aktuellen Woche zur Verfügung und überprüft, ob von der SV-Schnittstelle neue Menus bereitgestellt werden.	Extern
2	News	Liefert Neuigkeiten inklusive einem Bild pro Neuigkeit. Ausserdem ist es möglich, via CMS die Nachrichten zu verwalten und über eine Push-Nachricht die Nutzer auf eine Nachricht hinzuweisen.	6, 7, 8
3	Events	Speichert Ereignisse, welche wie Neuigkeiten gepusht werden können. Wie bei den anderen Services wird bei Änderungen am Inhalt via CMS das Token beim Security Service geprüft.	6, 7, 8
4	Study Courses	Dieser Service liefert eine Liste von Studiengängen inklusive einem Bild pro Studiengang. Auch hier besteht die Möglichkeit der Push-Notifikation.	6, 7, 8
5	Contacts	Als einfachster Service wird hier nur ein in HTML-formatierter Text mit Kontaktdaten geliefert, welcher bequem über das CMS bearbeitet werden kann.	6, 7, 8
6	Security	Authentifizierung der CMS-Anfragen.	—
7	Devices	Registriert Geräte für Push-Notifikationen und verschickt Notifications.	Extern
8	Images	Speichert und veröffentlicht hochgeladene Bilder.	—
9	Statistics	Liefert Statistiken.	1, 2, 3, 4

Tabelle 2: Liste der Microservices und ihre Abhängigkeiten

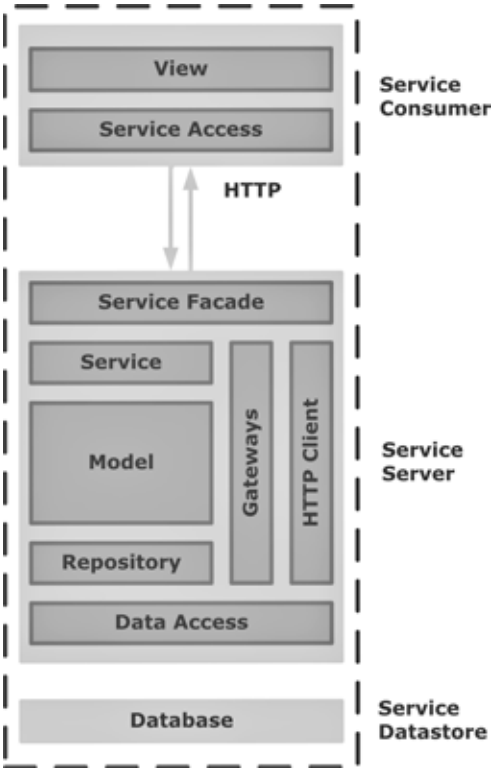

Abbildung 2: Konzept eines Microservices basierend auf einer Schichten-Architektur und HTTP Client als Gateway für die Kommunikation mit anderen Microservices.

	HTTP	URL	Beschreibung
1	GET	/	Returns a list of event objects.
2	GET	/page?page=nr	Returns a list of event objects corresponding to the given page number.
3	GET	/archive?page=nr	Returns a list of expired event object corresponding to the given page number.
4	GET	/{id}	Returns the event object with the given id.
5	POST	/	Creates a new event object.
6	PUT	/{id}	Updates existing event object with given id.
7	DELETE	/{id}	Deletes existing event object with given id.
8	GET	/push/{id}	Pushes event object with given id to all registered devices.
9	GET	/statistics	Returns a statistics representing the download of the event objects.

Tabelle 3: REST-API des Events-Microservice

teressierte finden in der Referenzdokumentation von *Spring Boot* [8] genügend Informationen, um die verschiedenen Aspekte der Implementation nachvollziehen zu können.

Wichtiger für das Verständnis der Gesamtapplikation sind die APIs, um zu verstehen wie ein Service Server von einem Service Consumer genutzt werden kann. Wie im Konzept dargelegt, stellen die Server-Services ein REST-API über die Service Facade bereit. Der Events-Microservice zum Beispiel implementiert ein API gemäss Tabelle 3.

Die Informationen, die zwischen *Service Consumer* und *Service Server* fliessen, werden als JSON Objekte codiert. Das Listing 1 zeigt beispielhaft die Implementation der REST-Methode #1 aus

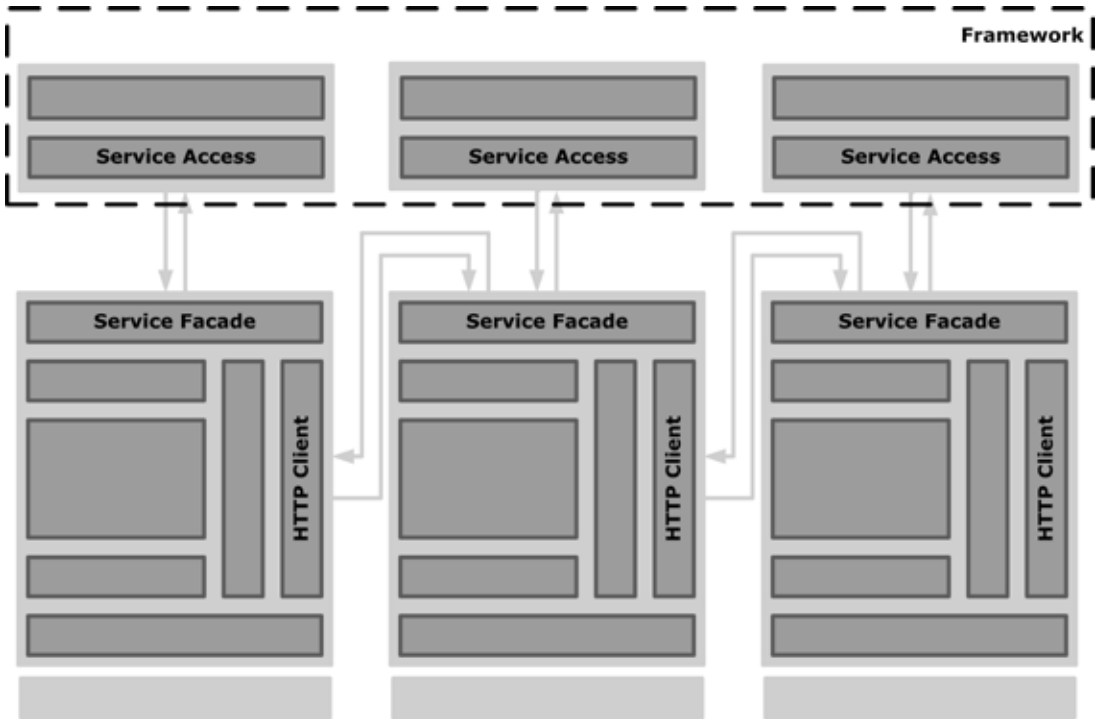

Abbildung 3: Konzept der gesamten Applikation, der Interaktionen und des Frameworks auf der mobilen Seite

```
@RequestMapping(value = "/", method = RequestMethod.GET)
public ResponseEntity<Map<String, Object>> getAllEvents() {                    #1
   Map<String, Object> resultObject = new HashMap<String, Object>();
   List<Event> events = eventsService.loadCurrentEvents();                     #2

   resultObject.put("events", events);                                         #3
   resultObject.put(ConstPool.TIME, Calendar.getInstance().getTime());
   resultObject.put(ConstPool.COUNT, events.size());
   return new ResponseEntity<Map<String, Object>>(resultObject, HttpStatus.OK);  #4
}
```

Listing 1: Auszug aus der Klasse EventsController, welche die Service Facade für den Events Service darstellt.

```
{
   "server_time":1447323320433,
   "count":16,
   "events":[
      {
         "id":53,
         "headline":"Infoveranstaltung",
         "subline":"Master of Science in Engineering…"
  …
```

Listing 2: Auszug aus der JSON Response

der Tabelle 3. In #1 wird als Antwort ein *ResponseEntity*-Objekt erwartet. Das Objekt besteht aus einer Map und dem HTTP Statuscode wie aus #4 ersichtlich ist. In #2 werden die Events aus dem Service *eventsService* geladen und anschliessend in die Map eingefügt (#3). Zusätzliche Informationen ergänzen die Antwort. *Spring Boot* wandelt diese Antwort in einen JSON-String um (siehe Listing 2), der von einem *Service Consumer* gelesen und verarbeitet werden kann.

Kommunikation unter Microservices

Microservices können auch die Dienste anderer Microservices nutzen. Deshalb muss ein Microservice die Möglichkeit erhalten, HTTP-Requests auf das entsprechende REST-API auslösen und eine JSON-Antwort gemäss Listing 2 behandeln zu können.

Aus der Tabelle 2 sieht man, dass zum Beispiel der Events-Microservice den Devices-Microservice nutzt, um über diesen Push-Notifikationen an die registrierten mobilen Geräte senden zu können. Diese Interaktion mit dem Devices-Microservice wird über einen REST-Request ausgelöst, wie in Listing 3 dargestellt.

In #1 wird die Event-Entität, die mittels Push-Notifikation angekündigt werden soll, über den Service aus der Datenbank gelesen. Als HTTP-Client kommt das REST-Template des Spring Frameworks zum Einsatz (#2). Die entsprechende Anfrage wird in #3 aufgebaut und an die URL *pushUrl* gesendet. Die Antwort des Devices-Microservices kann anschliessend überprüft werden, um daraus eine entsprechende Response an den Aufrufenden zu generieren.

```
@RequestMapping(value = "/push/{id}", method = RequestMethod.GET)
public ResponseEntity<Map<String, Object>> push(@PathVariable("id") Integer id) {
   Map<String, Object> res = new HashMap<String, Object>();
   Event event = eventsService.getById(id);                                    #1
   RestTemplate restTemplate = new RestTemplate();                             #2
   HashMap<String, Object> obj = new HashMap<String, Object>();

   obj.put("moduleId", "events");
   obj.put("contentId", event.getId());
   obj.put("title", "Events");
   obj.put("message", event.getHeadline());
   ResponseEntity<Void> response = restTemplate.postForEntity(pushUrl, obj, Void.class);    #3

   if (response.getStatusCode() == HttpStatus.OK) {
      res.put(ConstPool.PUSH_DATE, now);
      return new ResponseEntity<Map<String, Object>>(res, HttpStatus.OK);
   } else {
      return new ResponseEntity<Map<String, Object>>(HttpStatus.INTERNAL_SERVER_ERROR);
   }
}
```

Listing 3: Kommunikation zwischen Microservices

Abbildung 4: Launcher Screen auf der mobilen App mit fünf Modulen. Ein Modul entspricht einem Microservice. Das Modul wird über ein Icon repräsentiert, das vom Microservice bereitgestellt wird.

Microservices auf dem mobilen Client

Microservices werden auf der Server-Seite dank Frameworks wie Spring Boot sehr effizient unterstützt. In unserem Projekt wollen wir das Konzept der Microservices aber bis in den mobilen Client weiterziehen. Der Microservice soll auch auf der mobilen Seite seine visuelle Repräsentation (siehe Abb. 4) und seine Funktionalität festlegen können.

Zudem muss auch der Zeitpunkt, wann ein Microservice in Betrieb genommen wird, flexibel gehalten werden können. Das hat zur Folge, dass die mobile Applikation eine Möglichkeit umsetzen muss, um Module laden und darstellen zu können, die zu einem späteren Zeitpunkt in Betrieb genommen werden. Um diese Anforderung realisieren zu können, müssen die folgenden zwei Bedingungen erfüllt sein:

- *Service Discovery:* Das REST-API eines Microservices wird über eine URL identifiziert und kann beliebig gestaltet werden, da der Microservice auf irgendeinem Server betrieben werden kann. Um den Microservice nutzen zu können, muss ein Client herausfinden können, wo sich der Microservice befindet und wie er angesprochen werden kann.
- *Dynamic Service Loading:* Die Funktionalität eines beliebigen Microservices ist bei der Implementation der mobilen Applikation nicht bekannt. Deshalb muss die mobile Applikation ein Framework bereitstellen, das ein zusätzliches Nachladen der entsprechenden Funktionalität zu einem späteren Zeitpunkt erlaubt. Das dynamische Hinzufügen neuer Funktionen in eine mobile Applikation wird aber von den Betreibern der App Stores sehr unterschiedlich unterstützt. Während Google diesbezüglich sehr offen ist, weist die Firma Apple in ihren Richtlinien sehr hohe Einschränkungen aus. So schreiben sie in [2] folgendes: “Apps that download code in any way or form will be rejected” und “Apps that install or launch other executable code will be rejected”.

Unsere Technik-App ist eine hybride App, die auf Basis der Web-Technologien HTML, CSS, JavaScript programmiert ist. Neue Funktionalität wird demnach als JavaScript Code nachgeladen, was auch bei jeder mobilen, modernen Website der Fall ist. Deshalb ist die Chance gross, dass Apple das *Dynamic Service Loading* von JavaScript Code nicht abweist – was sich auch als wahr erwiesen hat.

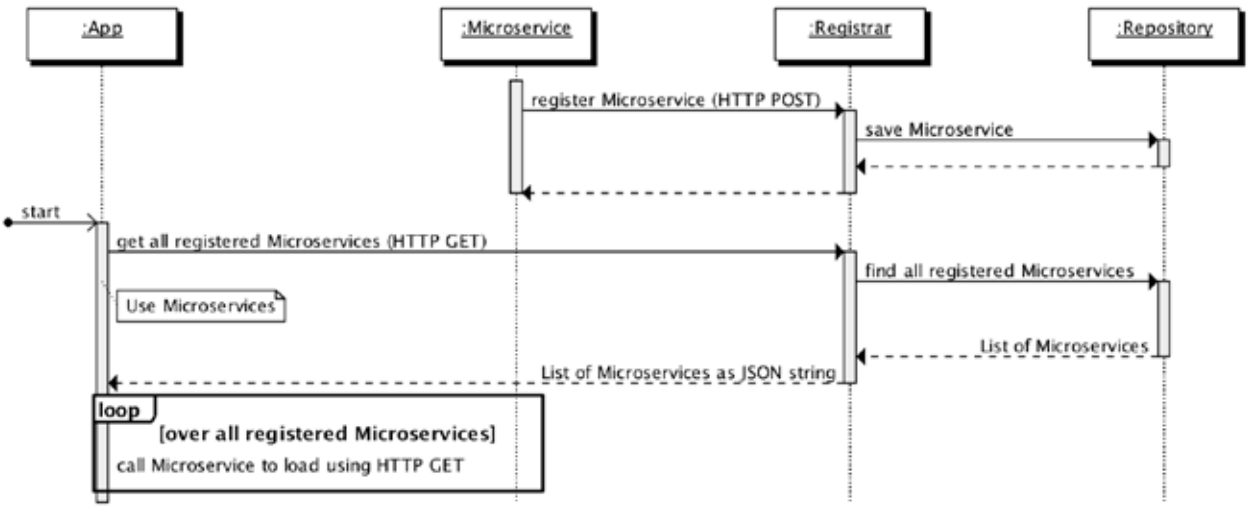

Abbildung 5: Das Sequenzdiagramm zeigt das Konzept für Service Discovery und die Interaktion verschiedener Aktoren (App, Microservice) mit der Registrierungsstelle Registrar

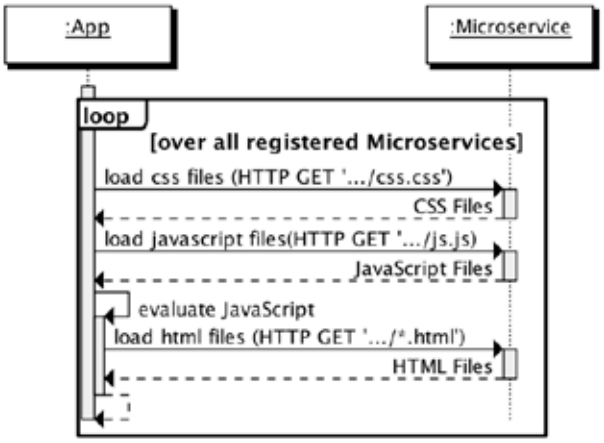

Abbildung 6: Sequenzdiagramm des dynamischen Ladens des mobilen Teils des Microservices

	HTTP	URL	Beschreibung
1	GET	/js.js	Der Code des Microservice für die mobile App, beinhaltet die Funktionalität auf der Client-Seite
2	GET	/css.css	Die CSS-Stylesheets für das Design des mobilen Microservices
3	GET	/icon.svg	Icon für den Launcher Screen
4	GET	/icon.png	Alternative Variante des Icons für den Launcher Screen, falls das SVG-Format nicht unterstützt wird
5	GET	/check	Statusabfrage

Tabelle 4: REST-API für das dynamische Nachladen

Das *Service Discovery* kann mit dem Design Pattern *Client-side Service Discovery* umgesetzt werden [7]. Dabei gibt es einen bekannten Ort, wo die Microservices registriert werden und wo ein Client die Zugangsinformationen zu den einzelnen Microservices abfragen kann. In Abbildung 5 ist der Ablauf eines Registrierungsschrittes und einer entsprechenden Abfrage in einem vereinfachten Sequenzdiagramm aufgezeichnet. Die Registrierungsstelle wird als Microservice implementiert und unter einer öffentlichen URL betrieben. Der Administrator dieses zentralen Microservices ist befugt, weitere Services zu registrieren und zu verwalten.

Für das *Dynamic Service Loading* müssen die Microservices eine einheitliche Schnittstelle umsetzen, so dass ein Client die fehlende Benutzerschnittstelle und die fehlende Funktionalität als HTML, CSS und JavaScript Files sowie Icons vom entsprechenden Microservice nachladen kann. Der Microservice selber muss diese Dateien als statische Ressourcen bereitstellen.

In Abb. 6 ist das Konzept dieses dynamischen Ladens abgebildet. Die aufgeführten REST-Abfragen legen das REST-API fest, welches jeder Microservice unterstützen muss, um die Integration in die mobile Applikation ermöglichen zu können. Über eine HTTP-GET Anfrage werden die fehlenden CSS- und anschliessend die JavaScript-Files geladen. Der JavaScript-Code wird sofort ausgeführt, was wiederum einen HTTP-Request auf den Server für zusätzliche statische Ressourcen wie HTML-Files oder Icons auslösen kann.

Um das *Dynamic Service Loading* unterstützen zu können, muss ein Microservice zusätzlich zu seinem API aus Tabelle 3 weitere Methoden öffentlich anbieten. Diese sind in Tabelle 4 zusammengefasst.

Der zentrale Code für das Nachladen ist in Listing 5 abgebildet. In Zeile #1 erfolgt der HT-

```
// Load and add CSS Styles
$http.get(url + 'css.css').success(function(data) {                    #1
   loadCSS(data);
});

// Load and execute JavaScript
$http.get(url + 'js.js').success(function(data) {                      #2
   eval(data);                                                         #3
});
```

Listing 5: Zentrale Funktionalität für das dynamische Laden von CSS und JavaScript-Code

```
/@RequestMapping(value = "/css.css", method = RequestMethod.GET)       #1
public Resource css() {
   return new ClassPathResource("style.css");                          #3
}

@RequestMapping(value = "/js.js", method = RequestMethod.GET)          #2
public Resource js() {
   return new ClassPathResource("controller.js");                      #4
}
```

Listing 6: Server Methode für das Laden des CSS und JavaScript Files

TP-GET Request auf die CSS-Files. Anschliessend kann der JavaScript-Code geladen (#2) werden. Dieser wird in der Zeile #3 ausgeführt. Daraus können weitere HTTP-Anfragen resultieren.

Listing 6 zeigt die entsprechenden Server Methoden. In Zeile #1 und #2 wird das Mapping auf die HTTP-Anfragen festgelegt und entspricht den Anfragen aus Listing 5. Das Laden der entsprechenden Dateien wird in Zeile #3 und #4 ausgeführt.

Fazit

Mit dem Umbau der App haben wir uns eine stark verbesserte Flexibilität im Umgang mit den einzelnen Modulen versprochen. Vor allem wollten wir die Möglichkeit erhalten, während dem Betrieb der Applikation „App4Technik" Module zu ersetzen, neue Module hinzuzufügen oder Module entfernen zu können. Mit der gewählten Microservice-Architektur ist es uns gelungen, diese Anforderungen erfolgreich umzusetzen.

Der Aufwand des Umbaus hat sich für uns aber nur deswegen gerechtfertigt, weil die Betreiber der App-Stores, insbesondere Apple-Store, das dynamische Nachladen vom JavaScript-Code unterstützen. Denn Microservice-Eigenschaften wie Unterstützung unterschiedlicher Technologien oder unterschiedlicher Programmiersprachen haben in unserem Kontext eher geringe Priorität. Das Studententeam hat aber zu Demonstrationszwecken Microservices in PHP implementiert und erfolgreich in das System integriert.

Da die Abhängigkeit der Microservices untereinander gering ist, kann die Kommunikationsinfrastruktur auf der Basis des einfachen HTTP-Request/Response-Modells aufgebaut werden. Deshalb ist die Komplexität eines einzelnen Microservices nicht hoch, einfach zu überblicken und gut verständlich. Fehlerbehebungen oder Ergänzungen können effizient realisiert werden.

Ein Microservice stellt in diesem Projekt ein kleines verteiltes Client-Server-System dar. Die Server-Komponente kann mit Hilfe geeigneter Frameworks wie *Spring Boot* sehr schnell implementiert und sehr gut mit automatisierten, funktionalen Tests validiert werden. Die Client Komponente hingegen wird im Kontext einer mobilen Applikation betrieben. Deshalb ist das Testen dieser Komponente anspruchsvoller und führt zu aufwändigeren Integrationstests, die nicht einfach zu automatisieren sind.

Für den Betrieb der gesamten Applikation „App4Technik" stehen verschiedene Optionen offen. Einerseits kann jeder Microservice über einen eigenen Webserver verfügen und in einem eigenen Prozess auf unterschiedlichen Servern laufen und andererseits können alle Microservices als Webmodule in den gleichen Webserver gepackt und über die Administrationsmöglichkeiten des Webservers separat administriert werden oder man wählt eine Kombination beider Varianten, was in unserem Betrieb aktuell der Fall ist.

Referenzen

[1] AngularJS: https://angularjs.org

[2] Apple. "App Store Review Guidelines": https://developer.apple.com/app-store/review/guidelines, 2015.

[3] Fielding, Roy. "Architectural Styles and the Design of Network-based Software Architectures", Dissertation, 2000.

[4] Fowler, Martin. "Microservices": http://martinfowler.com/articles/microservices.html, 2015

[5] Ionic: http://ionicframework.com

[6] PhoneGap: http://phonegap.com

[7] Richardson, C. "Client-side service discovery": http://microservices.io/patterns/client-side-discovery.html, 2014.

[8] Spring Boot: http://projects.spring.io/spring-boot

Secure Physical Access with NFC-enabled Smartphones

This paper presents a smartphone-based physical access control system in which the access points are not directly connected to a central authorization server. The access points ask the mobile phone whether a particular user has access or not. The mobile phone then relays such a request to the access server. The authentication of the smartphone is based on public-key cryptography. This requires that the private key is stored in a secure element or in a trusted execution environment to prevent identity theft. In our solution we use the following secure element archiectures: Host Card Emulation (HCE) and a microSD-based secure element. We show that the HCE approach cannot solve the relay attack under conservative security assumptions and we present and discuss an implementation based on a microSD secure element that still allows the access points to connect to the authorization server upon every access albeit the access points are not connected with it.

Christof Arnosti, Dominik Gruntz, Marco Hauri | dominik.gruntz@fhnw.ch

The smartphone has become the central gadget in our life and makes our wallet more and more redundant. We use the phone for mobile payment, for mobile ticketing and soon it will replace all the smart cards we carry in our wallet.

This paper[1] presents a physical access control system (PACS) in which the access card is replaced by a smartphone. The access points are not connected; they check whether access is granted by sending an access request to the mobile phone. The mobile phone either forwards such a request to the access server or, if it is not connected to the internet, presents an access token stored on the mobile phone to the access point. Access is thus possible even if the mobile phone is not connected to the internet.

The communication between the mobile phone and the access point is based on Near Field Communication (NFC). NFC is a short-range wireless technology that enables communication between a smartphone and an access point over a distance of approximately 10 cm or less. NFC operates at 13.56 MHz and can achieve (theoretical) transfer rates up to 424 kbit/s. A growing number of smartphones are equipped with NFC [12].

We define a PACS as a system that controls access to physical resources like buildings, rooms or protected areas, using user-specific access control rules. A PACS supports two main activities: authentication and authorization. When a person requests access to a physical resource then it claims its identity and the authentication process verifies this claim. Authorization then defines whether access is granted for this identity or not. In a PACS, the authentication process checks whether the person

- knows a secret (e.g. a password or a PIN),
- has something (e.g. a key or a token), or
- has a unique property (e.g. biometric properties).

PACS based on NFC are not new. Such systems typically store the access rights on the SIM card or on an embedded secure element and they depend on third party suppliers, for example mobile network operators (MNOs), trusted service managers (TSMs), smartphone manufacturers like Google, Apple or Samsung, or identity service providers like Legic IDConnect.

In this paper we present a PACS which is independent of third party providers. The smart card is replaced by the smartphone which is connected to the access server. However, since these phones are programmable and network connected, they provide a large surface for attacks [8]. In this paper we describe and analyze the security of different authentication solutions developed in the context of a concrete PACS. In particular we present a novel authentication process which prevents software proxy attacks.

The rest of the paper is organized as follows: In the next two sections we describe the general structure of our solution and the designed protocols. The security properties are analyzed in the section „Attack Surfaces“ and a solution to the authorization problem is presented in the section „Separate Secure Element“. Finally, we describe further (physical) attacks and compare the different PACS approaches. After an overview of related approaches we conclude with the results.

PACS Models

A classical PACS consists of three components: A server, identity cards and access points (doors with electronic components). There exist two common interaction models for these components.

In the *online access point* model the access point is connected with the server over a network connection. In this model, the card acts as an au-

1 This is the submitted version of a paper which has been accepted for the 13th International Conference on Advances in Mobile Computing and Multimedia (MoMM15), 11 - 13 December 2015 in Brussels, Belgium.

thentication-only component. The access point authenticates the card and accesses the server to get authorization information for the authenticated card. The server then decides whether the access is permitted or not.

In the *offline access point* model the access point has no connection to the server. Authorization data for a set of access points is stored on the card. Such authorization data contain access rights which are only valid for a limited time (which is defined individually by the server for each access point and each card) and has to be renewed periodically by the end-user. The access point is able to authenticate the card and to get the authorization data directly from it.

A PACS in which the identity cards are emulated by smartphones can follow both models. But since a smartphone is a connected device, a third model is possible, namely a model where the offline access point is connected to the server using the smartphone as a proxy. This model combines the advantages of the other two models, i.e. the access points don't need a network connection (which means reduced infrastructure costs) but nevertheless support the verification of access rights at access time.

In our system the smartphone is responsible for authentication. Authorization data is requested from the server by the smartphone after the initial contact between the access point and the smartphone (see Figure 1).

We also implemented the possibility to store access rights on the smartphone (similar to the *offline access point* model), but we do not discuss this in this paper since the security implications do not change.

Protocol Design

As shown in Figure 1, the protocol scheme of our smartphone-based PACS consists of two protocols: the authentication protocol to authenticate the smartphone to the access point and the authorization protocol to transmit authorization data between the server and the access point, using the smartphone as a relay.

The authentication protocol is based on public-key cryptography. On the smartphone, an RSA key pair is stored. The public key of this key pair is known to the authorization server and is included in the authorization answer which is signed by the server. This signature links authorization and authentication by proving that the authentication response is made by the device the authorisation response belongs to.

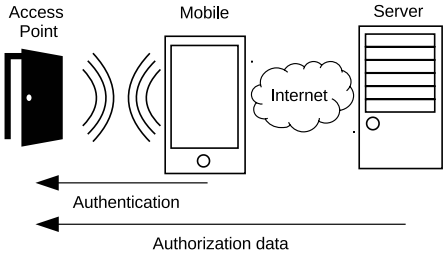

Figure 1: Smartphone solution

Both parts of the protocol are initiated by the access point and the requests for both parts are simultaneously sent from the access point to the smartphone using NFC. The answers are also simultaneously sent by the smartphone to the access point.

Authentication Protocol

To authenticate the smartphone (M), the access point (A) sends a nonce to the smartphone. The smartphone signs this nonce and sends the signature (sig) back to the access point. The access point then can check if the signature was created using the private key belonging to the same key pair as the public key sent and signed by the server in the authorization protocol.

1. NFC: $A \rightarrow M: Nonce_A$
2. NFC: $M \rightarrow A: sig_M(Nonce_A)$

Authorization Protocol

The authorization request sent by the access point (A) to the smartphone (M) consists of two segments: The access point ID and a nonce. This information is relayed to the server (S). By using SSL for the transport, the ID of the smartphone is provided to the server in the form of an X.509 client certificate.

The server can decide whether the access request should be granted or denied based on the access point ID, the smartphone ID, current time and access rights stored and managed by the server.

1. NFC: $A \rightarrow M: ID_A \,||\, Nonce_A$
2. Internet (TLS): $M \rightarrow S: ID_A \,||\, Nonce_A \,||\, Clientcert_M$
3. Internet (TLS): $S \rightarrow M: AccessOK \,||\, sig_S(Pubkey_M \,||\, ID_A \,||\, Nonce_A \,||\, AccessOK)$
4. NFC: $M \rightarrow A: Pubkey_M \,||\, ID_A \,||\, Nonce_A \,||\, AccessOK \,||\, sig_S(Pubkey_M \,||\, ID_A \,||\, Nonce_A \,||\, AccessOK)$

The access point ID, the nonce and the information whether the access was granted is encoded in the authorization answer, together with the public key of the key pair stored on the smartphone. The access point can verify the identity of the phone with the public key, and it can validate the authorization answer by means of a cryptographic signature based on a common secret known by the access point and the server.

To reduce the amount of data transferred between the server and the smartphone, all data already known by either side (access point ID, nonce and smartphone public key) are omitted. These

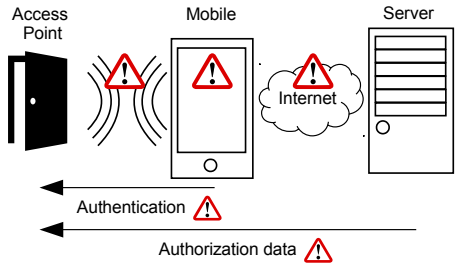

Figure 2: Attack surfaces: only the access point and the server assumed secure

data are re-attached by the smartphone before the whole packet is relayed back to the access point.

Attack Surfaces

Access control systems are security sensitive systems, a breach of security could lead to physical break-in and subsequently theft, sabotage or espionage. Because of this, security assumptions have to be conservative.

We assume that the smartphone and data transmission channels are insecure (marked with an insecure sign in Figure 2). Attackers with the right infrastructure can create man-in-the-middle scenarios to monitor or manipulate data sent over the internet or over the NFC connection. Malware inadvertently installed on a smartphone allows an attacker to gain full control of the installed software, to examine stored data and to run applications.

To allow the access point to securely decide about granting or denying access, two conditions have to be met: First, the authentication of the smartphone has to be secure; and second, the authorization data transmitted from the server must not be tampered with.

In the next two sections we describe attacks directly related to the protocol. Further attacks are described in the section „Physical Attacks“.

Attacking Authorization

An attacker with control over the network connection between the smartphone and the server can read and manipulate any transmitted data. To mitigate this type of attack, a TLS secured connection is used. X.509 certificates are used to authenticate the smartphone and the server to each other. With this technology in place, an attacker can only read the encrypted data, and any manipulation would immediately be detected.

The authorization data sent by the server to the access point is relayed by the smartphone, thus an attacker with control over the smartphone software can read and alter the transmitted data while it's passing through the smartphone. To allow detection of altered authentication data the server and the access point share a common secret which is used to digitally sign the transmitted data. By reading the authentication data sent by the server, the only valuable additional information an attacker gains is whether the attacked smartphone has access rights to the access point at the time of transmission.

With the nonce which is sent to the server and back to the access point, a replay attack (in which an attacker sends the same answer multiple times to reuse old authorization data) can be detected. The access point simply compares the received nonce with the sent one to see if the answer correspondents to the current request. To check the integrity of the authorization answer, the access point can create a signature of the payload data and compare it to the signature sent by the server. If the signatures don't match, the message was either not signed by the correct server, or changed in transit.

An attacker with control over the smartphone software, and thus over the TLS authentication used to verify the identity of the smartphone to the server, can send multiple authorization requests to the server to gain information about the access rights of the attacked smartphone.

Attacking Authentication

While detecting manipulation of authorization data is relatively easy since both ends of the communication are trusted components, the authentication process is more difficult to handle. The security of the authentication protocol relies on the possibility to store a secret key in a secure storage on the smartphone.

When confronted with a simple smartphone software solution, an attacker with system-level access can simply read the private key which is used to authenticate the smartphone against the access point. This private key can then be used on another device to impersonate the attacked smartphone.

Starting with version 4.3, Android provides support for hardware-based key stores [1]. Such key stores depends on the availability of security hardware (mostly integrated into the CPU of the smartphone) and on the software implementation of device manufacturers. If a hardware-based key store is present in a device, a smartphone application can use it to securely store the private key of a key pair and to execute private key operations without the possibility that the Android OS or any

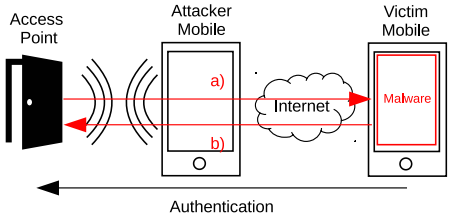

Figure 3: Software relay attack: a) the authorisation request, b) the authorisation answer

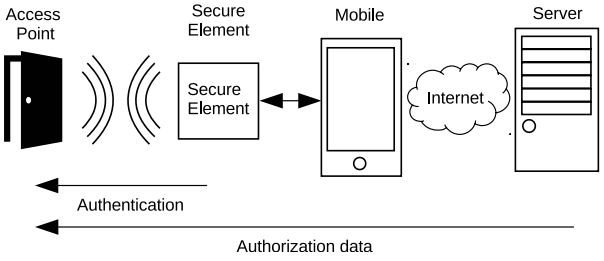

Figure 4: Use of a secure element for authentication

third-party applications can extract the private key [6, pp. 178-180].

Even if a hardware-based key store is used, an attacker can apply a software relay attack on the key store (similar to the relay attack on a secure element described by Michael Roland [14]) to execute the needed private key operations on the victim's smartphone at the time of access with a second smartphone (Figure 3). To achieve this goal, the attacker uses a smartphone to create a connection with the access point system. He then sends the authentication request by internet to a malware application on the victim's smartphone, where the malware can execute the necessary private key operations to generate the signature needed for authentication and send back the result to the attacker's smartphone.

The software relay attack can only be solved using a separate trusted processing environment with its own NFC connection to securely authenticate the smartphone to the access point. This processing environment executes all private key operations and sends the result directly to the access point without the possibility that any code executed on the smartphone CPU can access the result. Such a solution which still supports online authorization is described in the next section.

Separate Secure Element

As discussed in the last section, a separate piece of hardware is needed which can communicate to the access point by NFC and to the smartphone (see Figure 4). Such a secure processing environment is typically called a secure element (SE) or trusted execution environment (TEE). By adding this piece of hardware, we gain another secure component in the system which can be used for authentication.

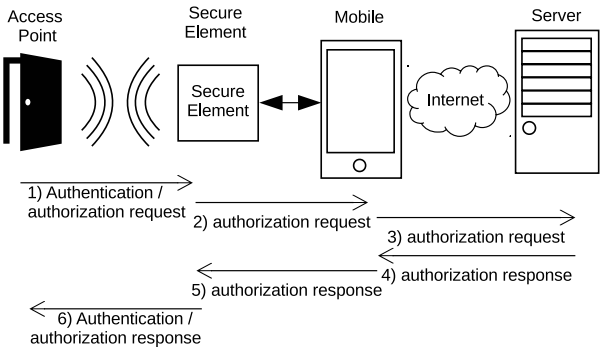

Figure 5: Overview of the protocol of a PACS with a microSD-SE

Secure Elements

Most modern SEs or TEEs are very small computers which are used in many different security sensitive applications – for example banking and credit cards, SIM cards, as an implementation for hardware-based key stores in Android phones, for access cards and also in smartphone NFC solutions. An SE contains at least a processing unit, program memory (typically flash memory), execution memory (RAM) and an interface to allow connections to other systems. Most SEs also contain cryptographic hardware to speed up the execution of cryptographic calculations. Typical interfaces to access the software of a smartcard are 8 pin plated contacts (banking cards), NFC (wireless banking cards) or soldering points (embedded SE).

While older secure element hardware was produced for special use cases, modern SEs contain an operating system with a standardized programming interface. An entity which wants to use secure elements – for example a bank – can develop an applet for their use case, deploy it to a number of secure elements, personalize the element (by executing special functionality of the applet to generate an ID or cryptographic secrets) and disable the programming functionality to guarantee data and application security. Only the issuer or a trusted third party can reprogram the secure element using cryptographically secured methods provided by the card operating system.

The most widespread programming interface for SEs is JavaCard. JavaCard is a slimmed-down Java variant specifically tailored to the security needs and low resources of a secure element. Special methods of persistence and transaction support are integrated in the language, and cryptographic methods are supported. Communication between a card terminal and the application on the card is standardized as ISO 7816. The protocol is a simple serial request/answer protocol which utilizes Application Protocol Data Units (APDU) to transfer information. A subset of these APDUs are standardized, others can be used in proprietary applications.

For our project we analyzed different programmable models of SEs which can be used in combination with a smartphone. One requirement was that the SE contains two separate interfaces, an NFC interface to communicate to the access point and an interface to communicate with the smartphone. Also, the application on the card must have the possibility to determine which communication channel is used to prevent the execution of the authentication method by software running on the smartphone CPU. We found such an SE in the form of a microSD card with a built-in NFC

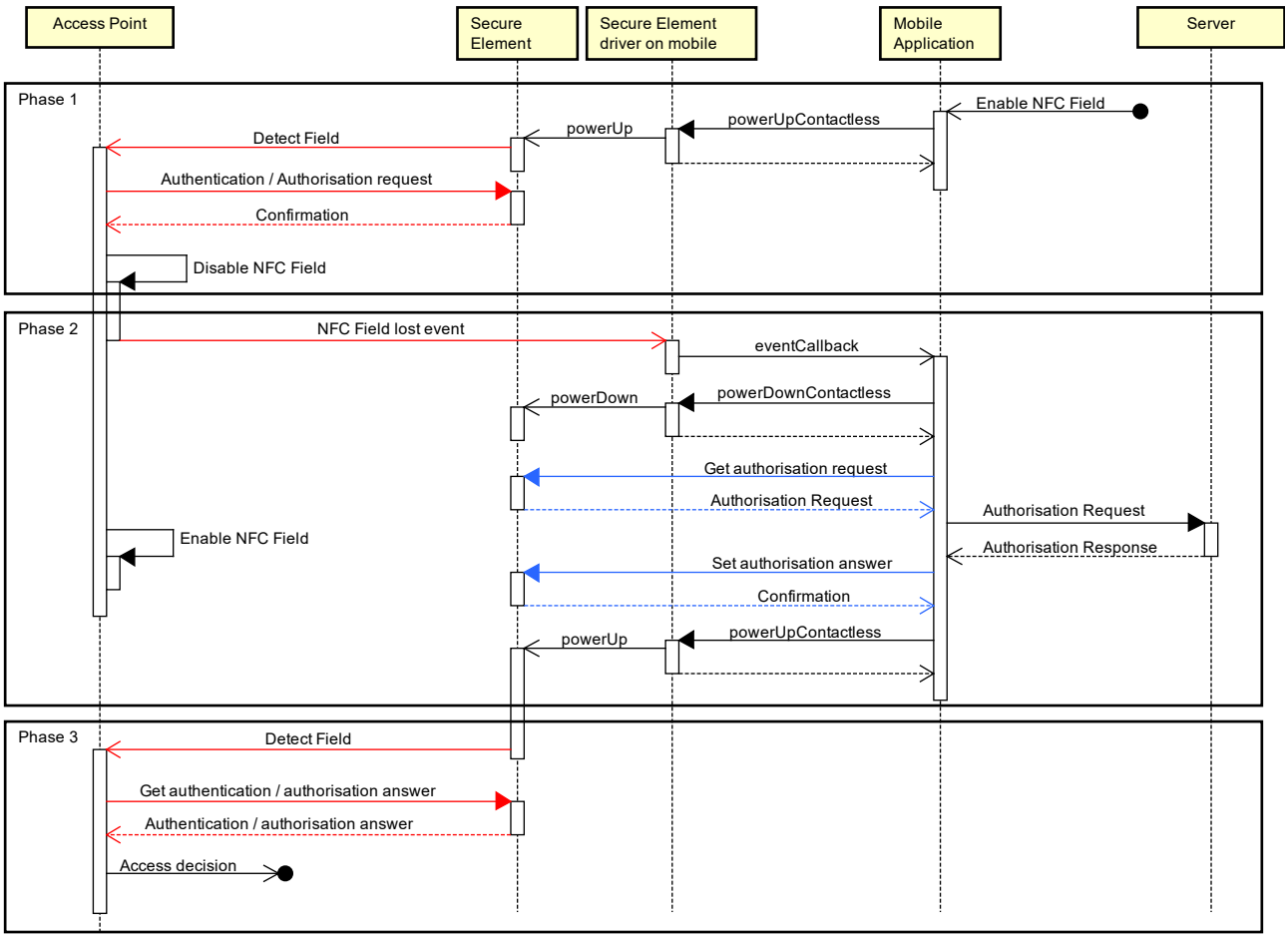

Figure 6: Sequence diagram of an access request using the secure element solution for secure authentication

transceiver: the CredenSE 2.10J developed by DeviceFidelity [3]. The SE embedded in this microSD card can be accessed from a mobile phone through a specific API.

PACS with a microSD-SE

In the PACS we designed and implemented we not only needed to authenticate the smartphone, but also at the same time to transfer authorization data from the server to the access point. To achieve this goal, we implemented a JavaCard applet which allows to authenticate the card to the access point as well as to relay data to the smartphone and subsequently to the server as shown in Figure 5.

All connections to the SE have to be initiated by either the smartphone or the access point, and it's not possible for the SE to communicate with both endpoints at the same time. Due to these circumstances we implemented a stateful JavaCard applet to relay the information to the smartphone and back. The NFC transceiver of the SE we used in the project has to be activated by an application running on the smartphone using a driver software. The driver software also allows to register a callback listener which gets notified when the secure element NFC transceiver enters or leaves the electromagnetic field of a NFC reader.

To use the PACS, the end-user has to activate the NFC transceiver of the SE by using the smartphone application which we developed. In the next few paragraphs we will describe the different phases of the process which is used to authenticate and authorize a phone to an access point. The transaction is also described as sequence diagram in Figure 6.

In the first phase of the process, after the smartphone user activates the NFC interface and touches the access point, the access point initiates the connection and sends the authorization and authentication request to the JavaCard applet which enters a second state. The access point then has to disable the NFC field to signal to the smartphone application that the first phase of the process is finished.

In the second phase of the process, the smartphone application initiates the connection to the SE after having received the callback that the NFC field of the reader is left (because the reader shut down the field). The smartphone application establishes a connection to the SE and sends a command to ask for the authorization request. While still holding the connection to the SE, this request is forwarded to the server by the smartphone application and – after having received the authorization answer from the server – the the answer is

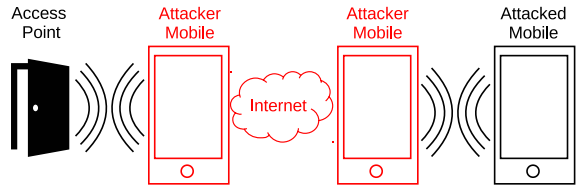

Figure 7: Hardware relay attack: An attacker uses two smartphones to extend the reach of the NFC transaction

sent back to the JavaCard applet which stores it and enters the third state. The smartphone now utilizes the driver software of the SE to enable the NFC transceiver and the access point gets a chance to detect the presence of the secure element.

In the third phase, the connection is again initiated by the access point. The access point can now read the answers for both the authentication and authorization requests sent in the first phase. The answer to the authorization request is the one sent by the smartphone in phase 2, the answer to the authentication request is computed by the JavaCard applet at this point in time. With these answers the access point has all necessary information to validate the authentication and authorization of the smartphone.

The authentication request is transferred by NFC directly from the access point to the SE. Since the SE is responsible for creating the authentication response and since the answer is directly transferred back to the access point by NFC, an attacker controlling the smartphone software cannot execute any public key operations nor read the private key. With the SE not allowing such operations by the smartphone, software relay attacks as described in the section „Attacking Authentication“ are not possible. By intercepting the connection between the access point and the secure element, an attacker with sufficient infrastructure can execute a hardware relay attack as described in the next section.

The transaction duration is increased by using this method because of the way the callback on NFC field events is implemented in the software driver. Internal in the driver, a loop checks the NFC field status every 0.5 seconds and notifies the callback listener on change of the NFC field status. We manage to trick the driver into performing this check every 0.1 seconds. With this change in frequency, the additional time compared to a software-only solution (thus, the maximal time needed until the callback is called plus the time needed for turning off and on the NFC transceiver of the secure element) is around 150 ms.

Physical Attacks

In this section we describe attacks where an attacker needs to gain physical access to the smartphone used in the PACS. There are two main attacks in this category: theft and the so called hardware relay attack [7]. Both of these attacks are also possible with a classic NFC card-based PACS.

To execute a hardware relay attack, an attacker needs two NFC-capable smartphones. These smartphones are connected – for example via Internet – and act as a proxy for NFC-transmitted data. With this setup, an attacker can extend the reach of the NFC transaction. To use this often-described attack [7, 10. 2. 11], the attacker places one of the smartphones at the reader, and the other on the victim's smartphone (see Figure 7). The smartphone placed at the access point now forwards all requests sent by the reader to the second attacker smartphone. The requests then get forwarded to the attacked smartphone and the answers are sent back using the same way.

When a smartphone is stolen, the attacker has the same possibilities as with the hardware relay attack (except that the relay infrastructure is not necessary). In our PACS, physical attacks and theft are addressed with the following two risk-reducing factors.

First, as the authorization data is loaded from the server at access time, an attacker cannot use a stolen smartphone after the theft was detected and the access rights got revoked server-side.

Second, the end-user needs to manually start the NFC transaction. For this, he has to unlock the smartphone and use a button inside of an installed application (but it's also possible to use a widget to place this button on the home screen of the smartphone). Because of the limited possibilities of the used secure element it's necessary that the SEs NFC interface is activated prior to a transaction, but the decision that this has to be manually done by the end user is a security feature. The end user (or the institution issuing the smartphone) can decide to configure security features like a display lock on the smartphone to complicate unauthorized use of the smartphone. If an attacker wants to perform a physical attack, he needs to activate the NFC interface of the secure element, and thus first needs to be able to operate the smartphone application.

However, if it is possible for an attacker to attack a smartphone at the same time physically and digitally, he can start an NFC transaction software-wise and use the NFC connection either in a hardware relay attack or directly at the access point. Such an attack works as long as it is not detected and the access rights are still valid on the server. To mitigate the effects of such an attack, a PACS needs to rely on additional authentication technologies beyond simply checking the ownership of a smartphone. Examples of such technologies are checks of biometric features like fingerprints or checks of knowledge like a PIN at the access point.

	Card	SE UICC or embedded	HCE software-only	microSD-SE SE & software
Third party independence (MNO/TSM/manufacturer)	(+) no dependency	(–) MNO/TSM dependency	(+) no dependency	(+) no dependency
Key security (security of key storage)	(+) secure	(+) secure	(0) device dependent	(+) secure
Hardware relay attack (theft / physical security)	(–) always as NFC is always on	(–) always as NFC is always on	(+) on unlocked device only	(+) after user interaction only
Software relay attack (software proxy)	(+) not possible	(+) not possible	(–) insecure	(+) secure
Time to open (usability)	(+) system dependent	(+) system dependent	(+) same performance as card	(0) like HCE + 150 ms
Online authorization (for online access points)	(–) not possible	(0) system dependent	(+) possible	(+) possible
Offine access rights (for online smartphones)	(–) terminal	(0) over MNO or terminal	(+) online over smartphone	(+) online over smartphone

Table 1: Comparison between different PACS

In the case that an attacker manages to perform a hardware relay attack he can read and manipulate the data sent in between the access point and the smartphone. The design of the protocol (as discussed in the section „Protocol Design“) guarantees that an attacker cannot manipulate the data undetected. Any data the attacker can read while eavesdropping this connection is either public, valid for only one transaction or of no added value. In particular the information whether access is granted for the attacked smartphone at the time of the interception has no added value as the attacker could simply watch if the physical access point allows access or not.

PACS Comparison

A classical PACS works with cards, but there exist systems where the card is emulated by a UICC-SE (SIM card) or an SE embedded in a smartphone. In this paper we have shown how such a smartphone-based PACS can be implemented independent of mobile network operators, the services of a trusted service managers and handset manufacturers, both with HCE and with a microSD-SE.

In this section we compare the following four PACS variants (these variants correspond to the columns in Table 1).

- *Card:* With this variant we refer to a card-only solution. Such a system could be based on Mifare DESFire cards which are ISO 14443-4 compliant and which are accessible over NFC.
- *SE:* This variant stands for all solutions which depend on third parties like MNOs, TSMs or handset manufacturers. Examples are the solution from Kaba [9] which is hosted by Legic Connect or Tapit, a solution from Swisscom [16]. Tapit will be denounced by the end of 2015.
- *HCE:* This is the solution described in the section „Protocol Design“ which is implemented without using a SE.
- *microSD-SE:* The microSD-SE-approach uses a separate SE to solve some security problems of a HCE-only solution as shown in the section „Separate Secure Element“. In our project we used a microSD-SE from DeviceFidelity.

We performed the comparison along the following criteria, and for all implementation variants we marked each criterion in Table 1 either with a + sign (positive), a – sign (negative) or with a 0 (neutral).

- *Third party independence:* Except for the SE-variant, all other variants are independent of a third party, i.e. the PACS service provider has full control over the technology and can provide its own applications.
- *Key security:* Under this criterion we compared how secure user credentials can be stored. For the HCE variant such credentials can be stored in a hardware-based key store if such a feature is provided by the phone hardware.
- *Hardware relay attack:* A hardware relay attack can be executed if the (emulated) card is accessible without further interaction. This is obviously the case for cards, but also for the SE-variant (for usability reasons, otherwise the access token could not be used if the phone has been turned off). For the HCE and the microSD-SE variants the hardware relay attack can only be executed if the device has been unlocked (and a special application has been started in addition for the microSD-SE variant). HCE on the other hand is vulnerable by this attack.
- *Software relay attack:* Obviously the card-only variant is immune to software attacks, and for the SE-variant the software relay attack is also not possible if the system is implemented properly and does not allow that private key operations are executed from the phone host.

Figure 8: Implementation of our PACS in action

This is the condition which is also met by the microSD-SE which we used in the implementation of our project.

- *Time to open:* We expect that all solutions show a comparable timing, except for the microSD-SE variant which takes about 150 ms longer than the HCE variant.
- *Online authorization:* The online authorization is possible for the HCE and the microSD-SE approaches. We don't know any systems where an UICC-SE or an embedded SE is used while also providing online authorization data via NFC, and we don't know if all requirements are met to enable such an implementation.
- *Offline access rights:* Although we focused on online authorization in this paper, our solution can also be used if the smartphone is not connected to the access server. In this case an access token is sent to the access point in response to an access request. These tokens are renewed regularly and automatically as soon as the smartphone is connected. For card-only systems access rights can be stored on the cards (in addition to the authentication credentials), but then the user has to reload this card at specific terminals. For the UICC-SE approach a reload of access tokens can be performed over a terminal or over MNO specific technologies. It would also be possible to load access rights to an UICC-SE using the smartphone's internet connection, but the SE would have to distinguish the access paths in order to prevent the software relay attack.

According to the criteria we used in this comparison the microSD-SE approach has a lot of advantages in the security and usability criteria. Financially, the microSD approach is relatively expensive since the cards need to be bought for every user. As the same infrastructure can be used for the HCE variant, the higher financial investment directly correlates to higher security.

Related Work

Several NFC-based access control systems for smartphones have been described and implemented, but most of them are not public. NFC-based PACS typically either use a UICC-SE [9,16] or they are HCE-based [17].

Before HCE was available in the Android framework, an alternative was to use the inverse reader mode [15]. Systems that adopted this approach are, for example, AirKey[2] from EVVA and NFC Porter[3] from IMA. These systems both store their credentials on the mobile phone for offline use [13].

Most UICC-SE solutions follow the *online access point* model described in the section „PACS Models", i.e. the access points are typically connected to the authentication server, and over these connected access points the data stored on the SE can be updated securely.

All HCE-based solutions suffer from possible software relay attacks. The same holds for all other access control solutions which store the credentials in a SE, but use other communication technologies like Bluetooth Smart (BLE) to connect to the access point and to the authentication server and thus use an application running on the smartphone to move authentication data. A system which follows this approach is HID's Mobile Access[4]. The relay attack problem is often mitigated by additional security checks such as the need to enter a PIN or the check of biometric features directly at the access point.

The general structure of our microSD-SE based solution follows the model described in [4], but that model explicitly excludes relay attacks as the focus is on delegatable authentication for NFC-enabled smartphones. To mitigate relay attacks distance-bounding techniques are proposed. These techniques determine an upper bound on the round-trip time of request-response pairs [5]. However, this approach cannot be applied to online solutions where the access server has to be connected before the response is sent back to the access point.

Results

We have presented a smartphone-based PACS in which the access points communicate with the access server over the smartphone connected to the access point. This relay approach allows different attacks, in particular the hardware- and the software relay attack. The hardware relay attack can be mitigated by protecting the smartphone with a screen lock.

2 http://www.evva.at
3 http://www.nfcporter.com
4 http://www.hidglobal.com

We have shown that the software relay attack can be prevented with a microSD-based SE which communicates directly to the access point. For the online authorization the microSD-SE must be able to communicate with the server over the smartphone. Our contribution is to show that such an approach can be implemented and that the speed is still acceptable. A picture of the implemented solution in action is shown in Figure 8.

A drawback of the microSD-SE approach beyond additional costs is that the microSD-card is typically provisioned by a single service provider (in our case this would be the provider of the PACS). An end user wanting to use microSD-SE based PACS from multiple service providers would have to switch microSD-cards.

A PACS usually has to support several levels of security. With the solution presented in this paper, the same infrastructure and the same protocols can be used for the HCE as well as the microSD-SE variants. The less secure HCE variant could be rolled out to most of the users who have access to a building, and users having access to high security areas inside that building could use the microSD-SE based solution.

Acknowledgements

We would like to thank the Swiss Commission for Technology and Innovation (CTI) which cofinanced this project. Many thanks also go to Carlo Nicola at FHNW for his help in the protocol design and to Markus Freund at Bixi for the vital task of implementing the access point part of the protocol.

References

[1] Android open source project. Android keystore system. https://developer.android.com/training/articles/keystore.html. Accessed: 2015-08-21.

[2] C. Arnosti and D. Gruntz. Man-in-the-Middle: Analyse des Datenverkehrs bei NFC-Zahlungen. IMVS Fokus Report, 8(1):24-31, 2014.

[3] DeviceFidelity Inc. CredenSE 2.10j classic is NFC card-emulation and certied JavaCard SE in a MicroSD. 2013. http://devifi.netfirms.com/devifi.com/assets/DeviceFidelity_CredenSE.pdf

[4] A. Dmitrienko, A.-R. Sadeghi, S. Tamrakar, and C. Wachsmann. SmartTokens: Delegable access control with NFC-enabled smartphones. In International Conference on Trust & Trustworthy Computing (TRUST), volume 7344 of Lecture Notes in Computer Science (LNCS), pages 219-238. Springer, June 2012.

[5] S. Drimer and S. J. Murdoch. Keep your enemies close: Distance bounding against smartcard relay attacks. In Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium, SS'07, pages 7:1-7:16, Berkeley, CA, USA, 2007.

[6] N. Elenkov. Android Security Internals: An In-Depth Guide to Android's Security Architecture. No Starch Press, San Francisco, CA, USA, 1st edition, 2014.

[7] L. Francis, G. Hancke, K. Mayes, and K. Markantonakis. Practical relay attack on contactless transactions by using NFC mobile phones. IACR Cryptology ePrint Archive, Report 2011/618, 2011. http://eprint.iacr.org/2011/618.

[8] T. Janssen and M. Zandstra. HCE security implications. Technical report, UL Transaction Security, 2014.

[9] Kaba. Mobile access solutions. http://www.kaba.com/en/kaba/innovation/654636/mobile-access-solutions.html.

[10] E. Lee. NFC Hacking: The Easy Way, 2011.

[11] M. Maass, U. Müller, T. Schons, D. Wegemer, and M. Schulz. NFCGate: An NFC Relay Application for Android. In Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks, WiSec '15, New York, NY, USA, 2015.

[12] NFC World. NFC phones: The denitive list. http://www.nfcworld.com/nfc-phones-list. Last updated on 21 August 2015.

[13] M. Roland and J. Langer. Comparison of the usability and security of NFC's dierent operating modes in mobile devices. e&i Elektrotechnik und Informationstechnik, 130(7):201-206, 2013.

[14] M. Roland, J. Langer, and J. Scharinger. Applying relay attacks to google wallet. In 5th International Workshop on Near Field Communication (NFC), pages 1-6, Feb 2013.

[15] C. Saminger, S. Grunberger, and J. Langer. An NFC ticketing system with a new approach of an inverse reader mode. In 5th International Workshop on Near Field Communication (NFC), Feb 2013.

[16] Swisscom. The swiss wallet of tomorrow. http://www.tapit.ch/en.

[17] Telcred AB. A new approach to access control. http://telcred.com. Accessed: 2015-08-20.

Wird Design Thinking erwachsen?

Design Thinking ist eine aus den USA stammende Methode, die sich in ihrem Selbstverständnis kompromisslos an den Bedürfnissen der Menschen statt an technischen Randbedingungen orientiert. In einem ersten Schritt geht es zunächst darum, diese Bedürfnisse der Nutzer zu verstehen. In kleinen interdisziplinären Teams werden anschliessend Lösungen entwickelt, wobei Einzelkämpfer und eine allzu kritische Haltung verpönt sind. Was zählt, ist der möglichst ungehinderte Fluss von Ideen, wobei auch die Sichtweise von fachfremden Personen geschätzt wird. Design Thinking besteht aus einer Vielfalt an Definitionen, Design-Artefakten, Ansätzen und Weiterentwicklungen. Dazu gehören Empathie mit den Benutzern, Prototypenentwicklung und Toleranz für den Misserfolg. Design Thinking kann ein Werkzeug sein, um eine agierende sowie flexibel organisierte Innovationskultur zu entwickeln. Vor allem dann, wenn ganzheitliche und nutzernahe Lösungen gefragt sind, um mit komplexen Aufgaben unserer Gegenwart umzugehen.

Thekla Müller, Christoph Stamm | thekla.mueller@fhnw.ch

In grossen Unternehmen wie Swisscom, Apple, Deutsche Bank, Google, Volkswagen, Deutsche Bahn, Siemens, AirBNB, usw. ist ein Wandel zu beobachten, bei welchem vermehrt das Design ins Zentrum einer Produkteentwicklung gesetzt wird. Bei diesem Wandel geht es nicht nur um die Ästhetik, sondern vor allem um die Anwendung der Grundsätze der Gestaltung auf das Arbeitsverhalten der Nutzer und Nutzerinnen[1] der Produkte. Die Unternehmen meinen erkannt zu haben, dass technische Überlegenheit oder eine hohe Qualität ihrer Produkte oder Dienstleistungen allein als Marktvorteil nicht mehr ausreicht, da sich Unternehmen überall auf der Welt dem immer wieder anpassen. Auf ihren Wegen zu neuen Innovationen setzen die oben genannten Unternehmen Design Thinking ein; ein Ansatz der den Mensch in den Mittelpunkt rückt.

In den folgenden Kapiteln zeigen wir, was *Design Thinking* ist und was eine nutzerzentrierte Innovationskultur bedeutet. Anschliessend gehen wir der Frage nach, ob Unternehmen, welche emotionale Benutzererfahrungen vor die Funktionalität stellen, Wettbewerbsvorteile erlangen und welche Konsequenzen das haben könnte. Es werden ausgewählte Design Artefakte sowie deren Einsatz in Unternehmungen vorgestellt. In einem abschliessenden Kapitel gehen wir auf einen wichtigen Aspekt in *Design Thinking,* nämlich das Scheitern ein.

Was ist Design Thinking?

Design ist historisch gesehen mit guter Gestaltung und Handwerk gleichgesetzt worden und Designer wurden als künstlerische Gelehrte zelebriert. Im Laufe der Weiterentwicklung des Designs als Vorgehensmethode bewegte man sich weg von der rein visuell und haptisch angenehmen Gestaltung sowie der Funktionalität neuer Produkte, hin zu einem umfassenderen Verständnis von Design: den Menschen und seine Fähigkeit, Gedanken und Prozesse durch multidisziplinäre Zusammenarbeit so zu entschlüsseln und greifbar zu machen, dass dabei Wege zu innovativen Geschäftslösungen entstehen können [VVA14]. Entstanden ist dieses Konzept im Umfeld der Produktdesignfirma IDEO und der *Stanford University*, wo 2005 das *Hasso Plattner Institute of Design* mit der ersten sogenannten *d.school* entstanden ist, um sich der Vermittlung der *Design Thinking* Methode zu widmen.

Design Thinking basiert auf der Annahme, dass Probleme besser gelöst werden können, wenn Menschen unterschiedlicher Disziplinen in einem die Kreativität fördernden Umfeld zusammenarbeiten, gemeinsam eine Fragestellung entwickeln, die Bedürfnisse und Motivationen von Menschen berücksichtigen und Konzepte entwickeln, die mehrfach geprüft werden [Bro06]. Deshalb wird ihr integraler Charakter und die dafür notwendige Auflösung gewohnter Begrenzungen oft als Spezialität dieser Methode genannt, wobei sich folgende drei Kernelemente gemäss dem Hasso Plattner Institut bewährt haben (siehe auch Abbildung 1):

- *Das Team wird gezielt multidisziplinär formiert*, um Ideen zu ermöglichen, die weit über die Fachgrenzen hinausgehen. Der Trend geht spürbar weg von einer auf die Entfaltung der eigenen Persönlichkeit ausgerichteten Haltung hin zur Wir-Kultur, zum gemeinsamen Erschaffen. Hier verspricht man sich das grösste Potenzial, da kollaborierende Teams schneller reagieren und ihre kollektive Intelligenz besser nutzen sowie nachhaltigere Arbeitsprozesse generieren und so auf erstaunliche Resultate kommen.
- Ideen entfalten sich am besten in einer *freien und flexiblen Arbeitsumgebung*. Variable Räume sind spontan auf die Bedürfnisse des jewei-

1 Zur Verbesserung der Lesbarkeit wird in diesem Artikel nur noch die männliche Form verwendet.

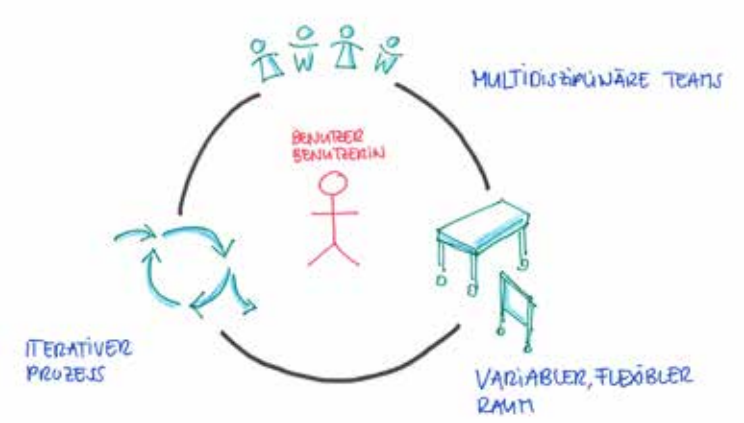

Abbildung 1: Die drei Kernelemente des Design Thinking

ligen Projektes anpassbar. Tische und Stellwände sind auf Rollen bewegbar. Wände und nahezu alle anderen Oberflächen werden frei genutzt, um Gedanken zu visualisieren und Arbeitsergebnisse zu teilen. Regale voll bunter Materialien laden dazu ein, Ideen schnell zu veranschaulichen und erlebbar zu machen.

- Mit dem *iterativen Innovationsprozess* navigiert das Team sich in den Lösungsraum. Der Prozess verlangt dabei nach einer offenen Fehlerkultur, da *Design Thinking* gerne an Unmöglichkeiten denkt, anstatt in Grenzen des Machbaren. Der Nutzer steht dabei mit seinen Motiven, Wünschen und Beweggründen im Mittelpunkt des empathischen Herangehens und Entwickelns. Der Prozess aktiviert dabei den analytischen sowie den kreativ-intuitiven Teil der Beteiligten.

Zielgruppe

Die Zielgruppe der Methode *Design Thinking* ist nicht auf Produkt-Designer, Vermarkter und Strategen beschränkt – sie soll jede Tätigkeit mit Kundenkontakt beeinflussen. Nehmen wir als Beispiel die Finanzbranche: typischerweise ist der häufigste Kontakt mit dem Nutzer durch Rechnungen und Zahlungssysteme, die für die interne Geschäftsoptimierung gestaltet oder als Kundenanforderungen festgelegt sind. Diese Artefakte sind wichtige Berührungspunkte. Es sind positive oder negative Eindrücke vom Unternehmen, welche der Kunde erhält. Wobei mehrere solcher Eindrücke unsere Einstellung gegenüber einer Unternehmung und dessen Bindung beeinflussen. In einer Kultur, die sich auf Kundenerfahrung konzentriert, werden solche Berührungspunkte rund um die Bedürfnisse der Nutzer und nicht um innerbetriebliche Effizienz ausgelegt. Dadurch wird der emotionalen Wahrnehmung der Nutzer mehr Rechnung getragen.

Was ist eine nutzerzentrierte Innovationskultur?

Bei *Design Thinking* steht der Nutzer mit seinen Bedürfnissen als Ausgangspunkt aller Überlegungen im Vordergrund. Der gesamte Prozess orientiert sich daran, Innovationen zu schaffen, die diese Nutzerbedürfnisse abbilden. Technische Machbarkeit und wirtschaftliche Rentabilität sind zentrale Bewertungskriterien, jedoch nicht Ausgangspunkt der Überlegungen. Ein weiterer zentraler Aspekt der Methode ist es, Nutzer intensiv im Kontext der Nutzung eines Produktes oder einer Dienstleistung zu beobachten und aus diesen qualitativen Beobachtungen Innovationsstossrichtungen abzuleiten und erkannte Nutzerbedürfnisse zu adressieren. Solche Beobachtungen generieren Daten zur Bedeutung des Produktes, der Dienstleistung und des Unternehmens, sowie Meinungen und Annahmen zum Thema. Somit unterscheidet sich das qualitative Vorgehen des Beobachtens im *Design Thinking*-Prozess deutlich von quantitativen Ansätzen, wie sie aus der klassischen Marktforschung bekannt sind.

Eine nutzerzentrierte Innovationskultur geht über das herkömmliche Rollenverständnis im Design hinaus und gibt dabei eine Reihe von Prinzipien an Personen weiter, welche Innovations-, Projekt-, Organisations- und Geschäftsideen in einem Kreativprozess erarbeiten wollen. Der iterative Innovationsprozess dargestellt in Abb. 2 orientiert sich an einer Kombination aus Verstehen, Beobachten, Sichtweise definieren, Ideen entwickeln, Prototypen erstellen und Testen. Die Darstellung entspricht dem Phasen-Modell, wie es an der *d.school* der *Standford University* und am Hasso-Plattner Institut der Universität Potsdam gelehrt wird und aus den Arbeiten von David Kelly hervorgegangen ist. Die einzelnen Phasen sollten dabei als Orientierungspunkte verstanden werden, die nicht streng linear durchschritten werden müssen, sondern zwischen denen das für das *Design Thinking* typische iterative Vorgehen stattfinden kann.

Design Thinking befindet sich in einem ständigen Weiterentwicklungsprozess, der von professionellen *Design Thinkern* und Organisationen angetrieben wird, die diese Methoden in verschiedenen Situationen und für unterschiedliche Her-

ausforderungen anwenden. Doch auch innerhalb der *Design Thinking* Literatur vertreten die verschiedenen Autoren und Lehreinrichtungen unterschiedliche Ansichten und Schwerpunkte. Dies spiegelt sich in den unterschiedlich zerlegten Prozessschritten wieder. Während Herbert Simon, Gavin Ambrose und Paul Harris sieben Schritte definieren, finden sich beim Hasso Plattner Institut sechs und bei Tim Brown und Robert Bauer drei Schritte [Pon14]. Prinzipielle Unterschiede sind jedoch kaum festzustellen, sie variieren lediglich in Beschreibung und Gewichtung hinsichtlich des Gesamtprozesses.

Emotionale Nutzererfahrung

Die nutzerzentrierte Innovationskultur befähigt Mitarbeitende, Empathie mit den Nutzern aufzubauen, indem sie deren Verhalten beobachten und Schlussfolgerungen ziehen über das, was Menschen primär wollen und allenfalls auch brauchen. Diese Schlussfolgerungen sind schwer in quantitativer Sprache auszudrücken. *Personas* beschreiben die Nutzer und *User Stories* oder *Scenarios* den Umgang mit Produkten aus der Sicht der Nutzer, wobei Wörter verwendet werden, die Sorgen, Wünsche, Sehnsüchte, Engagement und Erfahrung betreffen. Neben den Produktanforderungen und dem objektiven Nutzen wird auch über die emotionale Resonanz einer Wertschöpfung debattiert. So preisen einzelne Autohersteller nicht einfach nur sichere, komfortable und schön designte Hochleistungsfahrzeuge an, sondern sie versuchen das Gefühl zu beschreiben, welches die Kunden bei der Nutzung eines Autos haben werden. Die Kraft des Autos wird dann intelligent, der Komfort einzigartig und das Design atemberaubend. Im Werbetext werden ursprünglich neutrale Produkte mit einer emotionalen Reaktion verknüpft. Man versucht damit den Konsumenten einen Zusatznutzen zu suggerieren, der das Produkt von anderen der gleichen Warengattung unterscheidet.

Wettbewerbsvorteile durch Innovationsintelligenz

Unternehmen, die *Design Thinking* einsetzen, gehen davon aus, dass wenn sie ihre Leistungen, Dienstleistungen und Produkte aus den Augen ihrer Kunden betrachten und bei richtiger Erkenntnisanwendung einen Wettbewerbsvorteile erlangen können. Kevin Clark und Ron Smith von IBM erklären diese Art des Denkens genauer und bezeichnen sie als *Innovation Intelligence,* welche sich in drei Arten von Intelligenz aufschlüsseln lässt [CS08]:

- Die emotionale Intelligenz wird als die Fähigkeit des Verstehens von Menschen und ihren sozialen und kulturellen Kontexten beschrieben.
- Die integrale Intelligenz ist die Fähigkeit, unterschiedliche Kundenbedürfnisse und die Realitäten wirtschaftlicher Ökosysteme zu ganzheitlichen Systemen zusammenzuführen.
- Die experimentelle Intelligenz ist die Fähigkeit, alle fünf Sinne des Menschen zu verstehen und zu aktivieren.

Apple ist eine der bekanntesten Firmen, welche sich die Erkenntnisse der Innovationsintelligenz zu nutzen macht. Deshalb erstaunt die wahrscheinlich kürzeste Headline nicht, die jemals im Nachrichtenmagazin *Der Spiegel* gedruckt worden ist: „i" – einfach nur ein Buchstabe, zentriert gesetzt und sonst nichts [Bet11]. Natürlich handelt es sich dabei nicht um irgendeinen Buchstaben. Nachdem der iMac, der iPod, das iPhone und das iPad die Computerbranche stark beeinflusst haben, das „i" steht für eine Technologie, die nicht einfach bestehende Systeme optimiert, noch schnellere Prozessoren und noch höher auflösende Bildschirme einsetzt, sondern die ganz neue technische Möglichkeiten eröffnet [HV14]. „i" steht für die konsequente Umsetzung der Idee, den Nutzer und seine Bedürfnisse in den Mittelpunkt zu stellen. Nicht vom Gerät her zu denken, sondern vom Menschen und der Art und Weise, wie er die Welt wahrnimmt. Kurz, „i" steht für jene

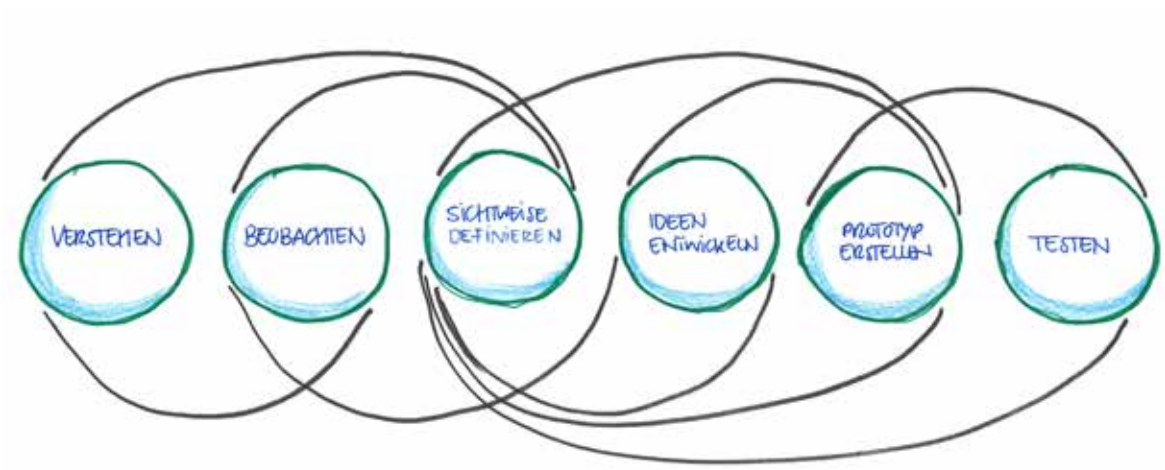

Abbildung 2: Iterativer Prozess in sechs Phasen

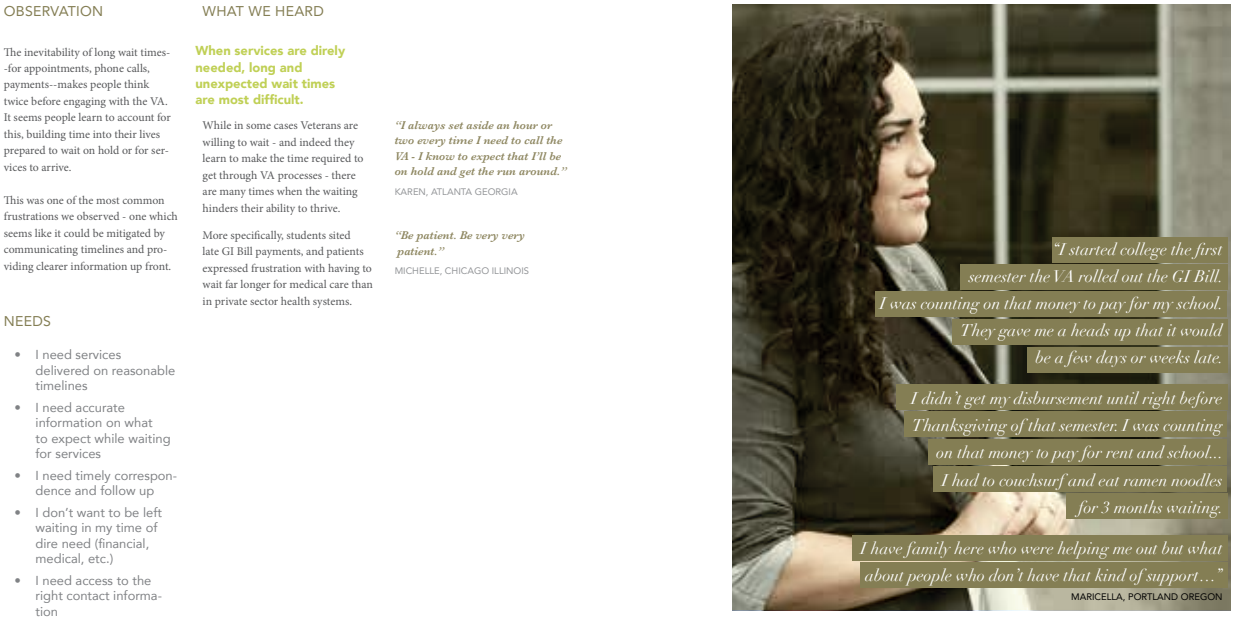

Abbildung 3: Research Findings aus dem Design-Artefakt Customer Journey Map des VA

ganz besondere Form innovativer Technologie, die so leicht und intuitiv zu benutzen ist, dass selbst Kleinkinder die eigentlich hochkomplexen Geräte spielerisch bedienen können.

Design-Artefakte

Design Thinking ist anfangs primär verwendet worden, um geeignete physische Objekte zu gestalten. Mittlerweile geht *Design Thinking* darüber hinaus und wendet sich komplexen, immaterielle Fragestellungen zu, wie z.B.: Welche Erfahrungen macht ein Kunde mit einer Dienstleistung? Unabhängig vom Kontext neigen *Design Thinker* jedoch dazu, physische Modelle, Prototypen, Skizzen oder generell Design-Artefakte zu erforschen, zu definieren und zu kommunizieren. Solche Design-Artefakte ergänzen und in einigen Fällen ersetzen sogar Arbeitsblätter, Spezifikationen und andere Dokumente, die in traditionell organisatorischem Umfeld definiert werden. Sie fügen damit eine weitere Dimension in der Erforschung der Komplexität hinzu, was nicht-lineares Denken bei der Bewältigung von nicht-linearen Problemen ermöglichen kann. Das folgende Beispiel soll diesen Aspekt verdeutlichen.

Das *U.S. Department of Veterans Affairs* (VA) verwendet ein Artefakt genannt *Customer Journey Map* [VA14], um die emotionalen Höhen und Tiefen der Veteranen in der Interaktionen mit dem Kundendienst des Department of Veterans Affairs zu verstehen. Diese Artefakte kommunizieren mithilfe von Geschichten und Erfahrungen der Betroffenen (Abb. 3), welche bei allen Beteiligten die Bereitschaft und Fähigkeit der Empathie fördern soll. Die enthaltenen Informationen geben darüber Auskunft, warum ihre Kunden das tun, was sie tun. Sie zeigen auf, wie ihre Erfahrungen mit der Organisation sind, wie die Interaktion zwischen den verschiedenen Berührungspunkten verläuft und wie man helfen kann, dass die Veteranen ihre gewünschten Ziele erreichen können. Die *Customer Journey Map* und andere Design-Artefakte helfen zu verstehen und präsentieren alternative Sichtweisen auf ein Problem.

Prototypen

Neben den beiden Aspekten der Nutzerzentrierung und der Inspiration durch Beobachtung und Befragung von Nutzern propagiert Design Thinking noch eine weitere zentrale Arbeitsweise, nämlich die Verwendung von Prototypen und Mock-ups als weitere Design-Artefakte zum Testen und Erläutern von Lösungsansätzen.

Design Thinker halten den Blick zunächst in die Zukunft gerichtet. Hier liegen die Vorstellungen, Träume, Möglichkeiten aber auch Unsicherheiten und Gefahren. Dabei treffen sie Annahmen, die sie durch Prototypen in die Gegenwart übertragen, um diese zu testen. Das Testen von Prototypen ermöglicht es Erfahrungen zu sammeln und Nutzungsmuster zu identifizieren. Damit verbinden sie intuitives mit analytischem Denken und bauen auf diese Weise eine Brücke zu analytischen Denkern. Diese Vorgehensweise hat zwei Gründe:

- Mock-ups helfen dem Team einzelne Konzepte zu erklären und zu evaluieren. Während abstrakte Beschreibungen häufig entscheidende Details auslassen oder verschleiern, zeigt ein grober Prototyp sehr schnell die Stellschrauben, an denen das Team ansetzen muss.
- Gemäss dem Grundprinzip *fail early and often* soll die angestrebte Lösung möglichst früh so konkret wie möglich und so aufwändig wie nötig dargestellt werden. So kann das Design-Team nicht nur besser arbeiten und kommunizieren, sondern auch Rückmeldungen der potentiellen Nutzer einfordern. Die tatsächliche Anwen-

dung durch Nutzer offenbart Schwächen in der Funktionalität oft schneller als eine theoretische Analyse. Gleichzeitig wird dieser Schritt des Testens auch genutzt, um die Akzeptanz einer neuen Lösung innerhalb und ausserhalb einer Organisation zu erkunden.

Nicholas Negroponte, Gründer des *MIT Media Lab*, formalisierte schon im Jahr 2000 den Akt des Prototypen-Erstellens in der Devise: *demo or die* [Nyt00]. Damit meint er, dass Ideen rein immaterielle, flüchtige und scheue Geschöpfe sind und nur durch *Prototyping* in etwas Handfestes überführt werden können. Daraus ergibt sich ein stetiger Strom von beeindruckenden Prototypen, welche die Innovationen in der Entwicklung optisch und zum Anfassen darstellen. Eine Demonstration hat im *MIT Media Lab* nur einmal zu funktionieren, da ihr alleiniger Zweck ist, Geld zu gewinnen, um die Arbeit im Lab weiterführen zu können. Auch sollen jene Unternehmen inspiriert werden, die Innovationen anzunehmen und für den realen Kontext weiterzuentwickeln.

Unternehmen welche eine nutzerzentrierte Innovationskultur leben, fürchten sich nicht, ihre gebastelten Ideen in einem öffentlichen Forum zu zeigen und neigen zu schnellen, iterativen Prototypen – eine Aktivität, die der Innovationsexperte Michael Schrage als *serious play* bezeichnet. In seinem Buch desselben Titels schreibt er, dass Innovation mehr kulturelle und soziale Werte als inhärente oder intrinsische Merkmale besitzt [Sch99]. Er fügt hinzu, dass *Prototyping* wahrscheinlich das pragmatischste Vorgehen ist, das innovative Firmen anwenden können. Auch belegen neuere neurologische Studien (vgl. beispielsweise [Mey01]) die grosse Bedeutung des haptischen Erlebens für das kognitive Verarbeiten. Prototypisierungen sind also nichts weiter als eine Reihe von Simulationen, bei denen Probleme antizipiert, Hypothesen getestet und Ideen beispielhaft ausgeführt werden, um diese in die Realität zu übertragen und eine Diskussion zu eröffnen.

Scheitern tolerieren

Spätestens beim Testen von Prototypen wird klar, dass durch das Feedback potenzieller Nutzer Überprüfungen und Überarbeitungen der Prototypen und somit möglicherweise auch der zugrunde liegenden Idee geleistet werden müssen. Zum Prozess des *Design Thinking* gehört also eine Offenheit gegenüber der Abfolge der Schritte und der Möglichkeit, mit Prototypen scheitern zu können. Die Methode lebt von einer absoluten Ergebnisoffenheit und somit auch von einer Kultur der Fehler, denn jeder Fehlschlag ist, wenn er früh erkannt wird, ein Gewinn für das Fortschreiten des Innovationsprozesses. Schliesslich können meistens hohe Entwicklungskosten gespart werden, wenn ein falscher Weg früh erkannt wird. Die sehr lebendige, da schnell durchgeführte, iterative Art des *Design Thinkings* kann somit auch konkreten finanziellen Nutzen mit sich bringen.

Eine Designkultur muss man pflegen. Es ist nicht besonders ermutigend, aber es ist in der iterativen Natur des *Design Thinkings*, dass man die Dinge nicht gleich zum ersten Mal richtig machen muss. Apple Inc. wird heute für ihre Erfolge gefeiert, aber wenn man etwas gräbt, entdeckt man Produkte wie das *Newton-Tablet*, das *Pippin* Gaming-System und das *Copland* Betriebssystem, welche nicht wirklich erfolgreich waren. *Pippin* und *Copland* wurden nach nur zwei Jahren wieder eingestellt. In diesem Sinn nutzt das Unternehmen ihr Scheitern um zu lernen und sieht es als ein Teil der Kosten für ihre Innovation an.

Für Alexander Grots, ehemaliger Geschäftsführer der IDEO Deutschland, gewinnt *Design Thinking* durch die Iteration an Wirkung und Effizienz, da die Grundregel *Früh-und-oft-scheitern* immer wieder neue Chancen bietet, um weiter zu lernen und die anfallenden Verbesserungen zu nutzen. Zudem ist er überzeugt, dass hohe Entwicklungskosten durch das frühzeitige Erkennen nicht erfolgreicher Ansätze gespart werden können [GP09].

Greg Petroff, Chief Experience Officer bei GE Software, erklärt, wie der iterative Prozess bei GE arbeitet. GE bewegt sich weg von einem Prozess der vollständigen Erhebung der Produktanforderungen. Teams erfahren erst während des laufenden Prozesses, was im Prozess zu tun ist. Mitarbeiter sollen erkennen, dass wenn sie soziale Risiken übernehmen, z.B. durch das Vorbringen von unausgegorenen Ideen, keinen Gesichtsverlust oder Straffauswirkungen zu befürchten haben.

Die erwähnten Unternehmen vertreten somit die Idee, dass der kreative Prozess und die Entstehung innovativer Ideen nicht durch Kritik gebremst werden und das Augenmerk auf einer ständigen Verbesserung der Ideen liegen soll. Weiter scheint es, dass neue Gedanken und verschiedene Betrachtungen zu ein und derselben Idee die Entstehung innovativer Lösungen fördert. Daher ist es vorteilhaft, mutige Ansichten vorzutragen und die Diskussion und Entwicklung einer Idee nicht durch voreilige Selbstkritik zu bremsen.

Function Follows Emotion?

In der Geschichte des Designs ist die Frage, ob die Form oder die Funktion Vorrang bei der Gestaltung geniessen sollte, häufig und kontrovers diskutiert worden. Der aus dem Funktionalismus stammende und vom Architekten Louis Henry Sullivan bekanntgemachte Ausdruck *form follows function*[2], schreibt die Abhängigkeit der Gestalt

2 Erstmals genannt wird der Terminus vom amerikanischen Bildhauer Horatio Greenough, der schon 1852 im Zusammenhang mit den organischen Prinzipien der Architektur von „form follows function“ spricht.

(äussere Form) von der Funktion bzw. dem Zweck fest und somit eine Unterordnung der Gestaltungsparadigmen. Das bedeutet aber nicht, dass rein funktionale Produkte angestrebt werden sollen, denn auch Ästhetik und Symbolik können bei Produkten des täglichen Bedarfs eine wichtige Funktion besitzen.

Der Ausdruck besass lange Zeit allgemeine Gültigkeit, wird aber seit den 70er-Jahren durch Abwandlungen wie *form follows fun*, *form follows emotion* oder *form follows fiction* in Frage gestellt. *Form follows emotion*, spiegelt den aktuellen Trend der Konsumindustrie wieder, den zunehmenden Forderungen der Nutzer nach individuellen Produkten gerecht zu werden. Heutzutage bevorzugen Nutzer Produkte, die ihre Individualität und ihre eigene Persönlichkeit zum Ausdruck bringen, bzw. mit denen sie sich identifizieren können. Ein anschauliches Beispiel dazu sind die i-Produkte der Firma Apple, die so gestaltet werden, dass sie bei ihren Nutzern positive Emotionen hervorrufen. Die technischen Kennzahlen und Werte und somit die reine Funktionalität verlieren dabei an Bedeutung. Dabei wird von der Überlegung ausgegangen, dass viele Konsumenten ihre Entscheidungen nicht aufgrund rein rationaler Überlegungen treffen, sondern eben auch aufgrund ihrer Emotionen. Positive Emotionen, die hervorgerufen werden, wenn man ein Produkt sieht, kauft, auspackt und verwendet.

Durch das Design ist es möglich, einem Produkt ein bestimmtes emotionales Profil zu geben, das sich von ähnlichen Produkten unterscheidet. Das heisst, das emotionale Profil hebt dieses von Konkurrenzprodukten ab und wird dadurch zu einem kaufentscheidenden Produktemerkmal. Es ist davon auszugehen, dass potentielle Kunden und Nutzer, die sich primär von emotionalen Profilen ansprechen lassen, sich selber eher auf der emotionalen und weniger auf der rationalen Ebene wahrnehmen. Aus der Sicht der Psychologie unterscheiden sich gerade Männer und Frauen im Erleben von Emotionen. So zeigen Frauen markantere emotionale Mimik, geben bereitwilliger über ihr emotionales Erleben Auskunft und erinnern emotionale Ereignisse besser als Männer [KG89]. Dass das iPhone also gerade bei Frauen sehr en vogue ist, kommt nicht unbeabsichtigt. Als der Designprofessor Paolo Tumminelle gefragt wurde, welches Geschlecht sich vom iPhone eher angezogen fühle, meinte er, dass mit den Rundungen und der glänzenden Empfindlichkeit der Oberflächen es eher Frauen seien. Die Dimensionen und die Form – im Prinzip ein Rechteck, seien aber männlich. Da das iPhone insgesamt aber nicht mechanisch geprägt ist, sei es nicht maskulin [TA10].

Design Thinking bedeutet unter anderem, dass der Nutzer mit seinen Wünschen und Motiven im Mittelpunkt des empathischen Herangehens und Entwickelns steht. Aber *den* Nutzer und *die* Nutzerin gibt es nicht. Die Bedürfnisse der Nutzer sind so unterschiedlich wie die Nutzer selber. Für die einen ist die Funktionalität das Allerwichtigste, anderen ist die Einfachheit das zentrale Kriterium und wiederum andere legen den primären Fokus auf Ästhetik oder Komfort. Durch eine offene Teamkultur und die Beobachtung von Nutzern beim Umgang mit Prototypen oder Konkurrenzprodukten soll vermieden werden, dass die Designer und Entwickler nicht einfach nach ihren eigenen Bedürfnissen designen und entwickeln, sondern ein Produkt schaffen, welches bei der anvisierten Zielgruppe Anklang finden wird. Je grösser und heterogener die Zielgruppe ist, desto mehr Kompromisse müssen in der Funktionalität gemacht werden und desto zentraler werden Anforderungen wie Einfachheit, Preis oder Qualität und gutes Design. Wo Einfachheit dominiert und Qualität schlecht wahrnehmbar ist, kann lediglich noch über Ästhetik und Emotionen eine Wertigkeit geschaffen und aufrecht erhalten werden, die von den Nutzern überdurchschnittlich hoch abgeglichen wird. In diesem Sinne kann also durchaus behauptet werden, dass wir uns momentan in einer Phase *function follows emotion* befinden.

Referenzen

[Bet11] Bethge Philipp u.a.: i, Der Spiegel Nr. 41, S. 68-77, 10.10.2011.

[Bro06] Brown Tim: Innovation Through Design Thinking, 2006, Vortrag an der MIT World. http://bit.ly/1ONNr3q

[CS08] Clark Kevin, Smith Ron: Unleashing the Power of Design Thinking, 2008, Design Management Review Vol. 19 No. 3.

[GP09] Grots Alexander, Pratschke Margarete: Design Thinking - Kreativität als Methode. Marketing Review St. Gallen, 26(2), 18–23, 2009.

[HV14] Hofmann Martin Ludwig, Vetter Andreas K.: Design Thinking, Das Denken, das Apple & Co. gross gemacht hat, 2014.

[KG89] Kring Ann M., Gordon Albert H.: Sex differences in emotion: Expression, experience, and physiology, 1998, Journal of Personality and Social Psychology.

[Mar09] Martin Roger: The Design of Business, Why Design Thinking is the next competitive Advantage, 2009.

[Mey01] Meyer Susanna: Produkthaptik, Messung, Gestaltung und Wirkung aus verhaltenswissenschaftlicher Sicht, 2001.

[NYT00] The New York Times, M.I.T. Media Lab at 15: Big Ideas, Big Money - Demo or die, 09.11.2000. http://www.nyti.ms/1Ww9SRH

[Pon14] Ponsold Stefan: Innovationsmanagement durch Design Thinking, 2014.

[Sch99] Schrage Michael: Serious Play: How the Word's Best Companies Simulate to Innovate, 1999.

[TA10] Tages Anzeiger: Die Gefühle, die Apple schafft, wirken wie eine Volksdroge, 28.07.2010, http://bit.ly/1m521dz

[VA14] Toward a Veteran Centered VA, Piloting Tools of Human-Centered Design for America's Vets, U.S. Dept. of Veterans Affairs' Center for Innovation. http://1.usa.gov/1jHpsbb

[VVA14] Vianna Maurício, Vianna Ysmar, Adler K. Isabel, Lucena Brenda, Russo Beatriz: Design Thinking, Innovation im Unternehmen, 2014.

Annular Barcodes

In this publication[1] we present a generic design for a novel annular barcode. On round media, circular or annular barcodes are more natural and preferable instead of traditional rectangular barcodes. Especially on round media with a distinctly convex or concave conic surface, an annular barcode is even more preferable than a full circle, because printing the barcode in the center of the media might be much more complicated or imprecise. Our generic annular barcode design supports different barcode and module sizes and therefore different data sizes. It includes different marker, synchronization, and data areas. For improved data robustness a data protocol and error correcting codes similar to them in QR codes are suggested. The feasibility of the design is shown by an implementation of an efficient and effective barcode decoder in C++ and a series of tests with distorted and noisy pictures of our annular barcodes.

Robin Brügger, Paul Glendenning, Christoph Stamm | christoph.stamm@fhnw.ch

A barcode is an optical machine-readable representation of data relating to the object to which it is attached [Wiki]. Originally barcodes systematically represented data by varying the widths and spacings of parallel lines. These may be referred to as linear or one-dimensional (1D). Later they evolved into rectangles, dots, hexagons and other geometric patterns in two dimensions (2D). Although 2D systems use a variety of symbols, they are generally referred to as barcodes as well. Barcodes are usually scanned by special optical scanners called barcode readers, but smartphones with built-in cameras and interpretive software can do the same work.

There are applications and situations where circular or annular barcodes are more natural and preferable instead of traditional rectangular barcodes, because of the round form of the barcode medium. Especially on round media with a distinctly convex or concave conic surface, an annular barcode is even more preferable than a full circle, because printing the barcode in the center of the media might be much more complicated or imprecise.

Circular Barcodes

Circular 1D barcodes fulfill the required annularity. They were extremely popular barcodes used by CD/DVD retailers for tagging their items [TR]. Figure 1 shows such a circular barcode on the left hand side, made up of a series of concentric circles and typically based on a standard barcode symbology like I2of5 (Interleaved 2 of 5). The barcode is thus readable by the same devices used to read traditional barcodes. Its payload depends mainly on its radius and is usually only a few decimal digits (2 digits in the example). Its data density (bits/area) is quite low, because of the large redundancy in all directions.

A ShotCode, depicted in Figure 1 on the right hand side, is one of the few existing circular 2D barcodes [BC09]. It was developed to share links and can be read with low resolution mobile phone cameras or webcams. In some application ShotCode is not suitable either because of its limited payload (5 to 6 bytes) or because of its used circle center. Today, ShotCode has been widely replaced by QR codes (Figure 2), because of its very limited payload.

Since we are interested in annular barcodes with high data density and have not found any such codes, we have designed a novel barcode format for codes with a data density comparable to QR codes. As a proof of concept we also implemented an encoder and decoder for our annular barcodes.

Requirements

In addition to the basic requirement of an annular form, there are further requirements for our barcode design:

- *Minimum payload:* The payload of the barcode should be at least 25 alphanumeric characters. Assuming extend 8 bit ASCII code as character code results in a minimum payload of 25 bytes or 200 bits.

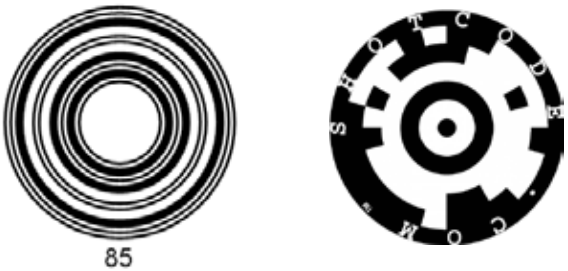

Figure 1: left) Circular Barcode (1D) [TR], right) Shotcode [Mud06]

1 This publication is based on Robin Brügger's bachelor thesis „Ring-shaped Barcodes", 2015, in collaboration with NANO4U AG.

Figure 2: QR-Code

- *Support for different sizes:* The barcode design has to support different sizes, from a few millimeters to a dozen of centimeters. In a barcode with a maximum outer diameter of 8 mm, for example, the smallest module size should be not less than 0.2 mm or 5 times the lateral resolution of the scanner.
- *High robustness:* Perspective distorted and noisy images of barcodes should be either correctly decoded or rejected. The usage of error correction codes is allowed as long as the minimum payload is guaranteed.

Annular Barcode Design

In this Section our novel annular barcode design for a code with a data density comparable to QR codes is explained in detail. For reasons of improved robustness in large barcodes two additional small adaptions are useful. The three design types differ only in the number of repetitions of the so called start marker and radial zebra patterns (Figure 3). Figure 4 shows a typical barcode of type 1 generated by our encoder (left), and the same barcode with its highlighted special purpose zones for recognition, perspective correction and timing (right):

- *Solid black ring*: for locating the barcode in an image. The black ring surrounds a white circle;
- *Square markers*: for a rough perspective correction. They are arranged in a perfect square;
- *Start marker*: defines the beginning in the otherwise continuous circle. It is also used to determine the outer radius of the barcode;
- *Angular zebra pattern*: determines the angles between the modules. It is also used to achieve an accurate perspective correction. Therefore, the angular module count is always dividable by 8. Thus, it is possible to select four modules of the angular zebra pattern which lay in a perfect square, facilitating the perspective correction calculation;
- *Radial zebra pattern*: determines the radial timing of the rings (how far apart the rings are) and the reading order (counter-clockwise or clockwise);
- *Data zone*: The data zone can be interpreted as a curved table, addressing modules by their polar coordinates. The addressing of the modules is organized in concentric circles, beginning immediately after the radial zebra pattern and from outside towards the inner and smaller modules near the solid black ring. The data zone is usually used for payload interpretation (header information), payload, and error correction codes.

The size and shape of the barcode are determined by four basic dependent parameters:

- Type 1-3: The type defines the number of repetitions of the start marker and radial zebra pattern;
- Minimal module size: The minimal module size $s_{\min}$ is the minimal width and height of a single module. It is also the thickness of the individual rings;
- Minimal radius: $r_{\min}$ is the distance from the center of the barcode to the beginning of the inner square markers. The square markers are the smallest module-sized structure in the barcode. $r_{\min}$ has to be clearly larger than two-times $s_{\min}$.
- Maximum radius: $r_{\max}$ is the distance from the center of the barcode to its outermost point. The maximum radius is dependent of the minimal module size and the minimal radius: $r_{\max} = r_{\min} + k\, s_{\min}$, where $k - 2$ is the number of rings in the data zone.

The concrete values of the last three parameters depend mainly on the resolution of the printing and scanning devices, the available area, and the payload requirements.

Figure 3: The three types of generic annular barcodes. Repeating start marker and radial zebra pattern increases robustness.

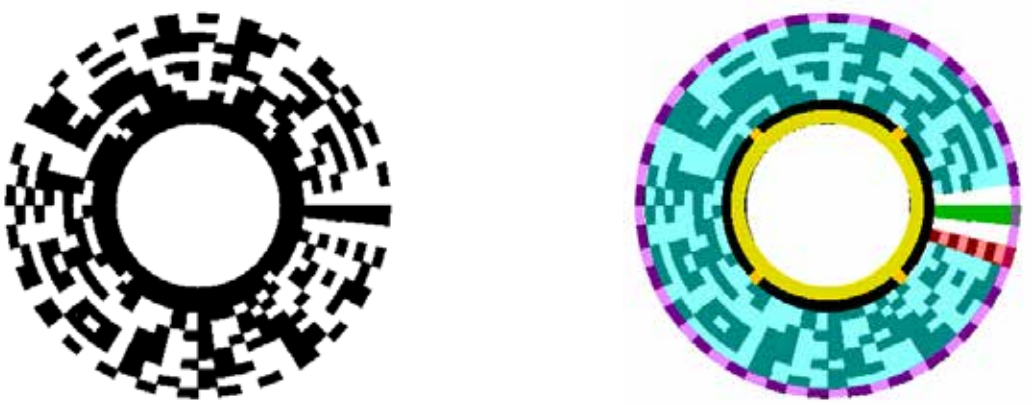

Figure 4: left) Annular barcode of type 1 with a data zone of 56 bytes; right) the same barcode but with highlighted special purpose zones (see also back inside cover for a color image)

Decoder

The barcode decoder is responsible for detecting, recognizing, and decoding a barcode in an image. Its input is an image of an annular barcode and its output is the data stored in the barcode. In case of a colored input image, it is first converted to a grayscale image and then binarized. For binarization the well-known method of Otsu usually results in a good binary image. In order to cope with noisy images, a Gaussian blur with a 5x5 kernel is applied prior to binarization.

The decoder cannot fully follow a simple process-chain of four stages („recognition", „perspective correction", „detecting timing information", „module evaluation"), because the angular zebra pattern (a part of the timing information) is also needed to achieve an accurate perspective correction. Therefore, the decoder steps „perspective correction" and „detecting timing information" are not strictly sequential. Furthermore, a two-step approach is used for the „perspective correction", resulting in the seven process stages depicted in Figure 5.

Each stage of the pipeline works on a best effort basis. If for example the recognition stage fails because it does not find the basic characteristics of our barcode design, the input image is discarded and the rest of the decoding-chain is not activated. However, if the recognition stage succeeds, then it returns the center of the barcode which is needed as input in the second stage. This is done by blob[2] analysis. General blob analysis allows filtering the blobs by area, circularity, inertia, convexity, color, and further criteria [Mal15]. Since the center of our barcode lies inside an empty white circle, we can search for white blobs and analyze them. To be a candidate for the barcode center, the white blob needs to have a circular shape or because of perspective distortion at least an elliptical shape. Circularity values between 0.8 and 1.0 are useful.

- The barcode's size lies between certain boundaries. If it is bigger than the field of view of the camera, it cannot be decoded because some information is not visible and therefore missing. On the other hand, it can be expected that the barcode has a minimum size which is a multiple of the smallest reasonable module size. Module sizes below 3 pixels are too small for robust decoding and hence ignored.

Detection of Square and Start Markers

In the second stage of our decoding-chain we try to detect the four square markers for rough perspective correction and the one to three start marker(s). In an unmodified barcode the start markers are black and the square markers are white. In a barcode without perspective distortion the four square markers lay in a perfect square.

Based on the observations that a start marker is the longest radially continuous stretch of black pixels, the algorithm uses a radial scanline revolving around the estimated center of the barcode (Figure 6). The scanline starts at 70% of the minor axis of the central ellipse (depicted by a green circle) and ends with the first pixel after the first black pixel. Along the radial scanline the length of the first continuous stretch of black pixels is evaluated. Figure 7 shows a possible plot of the first continuous black distance at different angles. The four smallest values signify the location of the four square markers, the maximum is where the start marker is located.

Because of the small discreet angular steps the scanline uses to revolve around the center, it is possible that two of the four lowest points belong to the same square marker. Errors are avoided by enforcing a minimum angular distance from one perspective marker to the next.

Sometimes the longest continuous black distance is not one of the start markers but an optical edge between black modules located at alternating positions on both sides of the edge. In order to avoid such errors, the found start markers are verified by checking that they are surrounded by continuous white pixels on both sides.

2 blob: binary large object

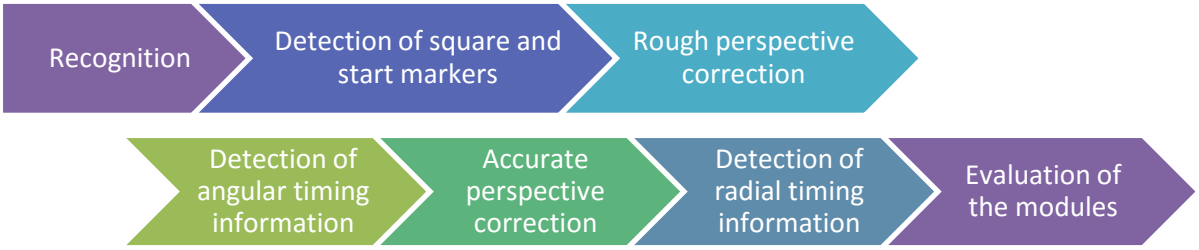

Figure 5: The seven stages of our decoding chain

The four square markers form a square in our barcode design. Due to the perspective distortion induced by the barcode scanner, the detected square markers in the input image may not form a square, but any quadrangle. Using a pseudo perspective transformation model for the distortion it is possible to compute a four-point linear transformation matrix T_1 between the positions of the detected square markers and their expected positions in a square. This transformation matrix T_1 is then applied on the entire input image. Further detection steps are performed on the transformed image.

The limited accuracy of this perspective correction is mainly due to the small size of the quadrangle formed by the square markers. Any inaccuracy in the location of the markers is amplified in outer areas of the barcode. Therefore, a more accurate perspective correction is applied during the next stage in the decoding-chain.

Detection of Angular Timing Information

The roughly undistorted image is the input of the fourth stage in our decoding-chain. In this stage the angular timing information has to be detected to determine the angle between two consecutive data modules. It also sets the foundation for a more accurate perspective correction which is necessary in the last stage. The angular timing information is stored in the zebra pattern around the outer edge of the barcode.

The angular zebra pattern is detected by a scanline revolving around the estimated center of the barcode (Figure 8). The scanline starts on a circle just outside the barcode and scans towards the center of the barcode trying to find the black modules. The distance to the occurrence of the first black pixel is evaluated. If this distance is within certain bounds around a value B, we have found a black module. An initial value of B is determined by the distance between the circle around the barcode and the outside of the start marker.

Although the input image has been roughly perspective corrected prior to this processing stage, the correction might not be perfect. B must therefore be updated at black modules in the angular zebra pattern to reflect the local situation. This is done by choosing B as the mean value of the already processed distances between the circle and the black module in the angular zebra pattern. Even with this adaption, detecting the angular zebra pattern occasionally fails. To prevent the decoder from progressing with wrong settings, the detected black modules are subject to verification.

Since the number of modules in the outer zebra pattern is dividable by eight, the number of black modules is dividable by four. Given an array with the position of every black module in the angular zebra pattern, selecting four of them lying in a square is easy. Such four positions are used to compute a second four-point linear transformation matrix T_2. This matrix T_2 performs a mapping between the roughly perspective corrected image and the accurately perspective corrected image. Since the aim is to avoid precision loss by doing two transformations on an image (first a rough correction T_1, than the accurate one T_2), the two perspective correction matrices T_1 and T_2 can be multiplied to allow transformation from the in-

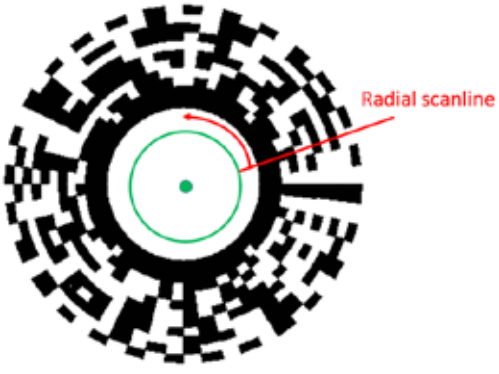

Figure 6: Radial scanline revolving around the estimated center of the barcode

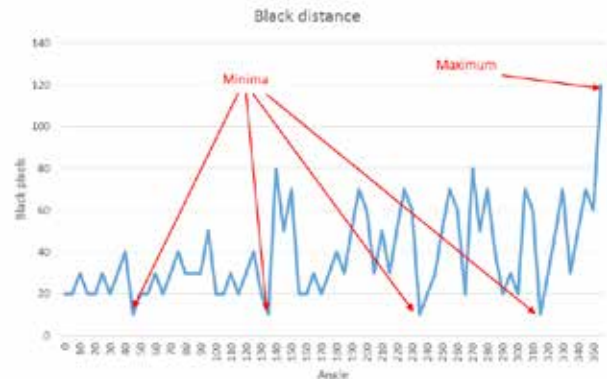

Figure 7: Plot of the continuous amount of black pixels at various angles

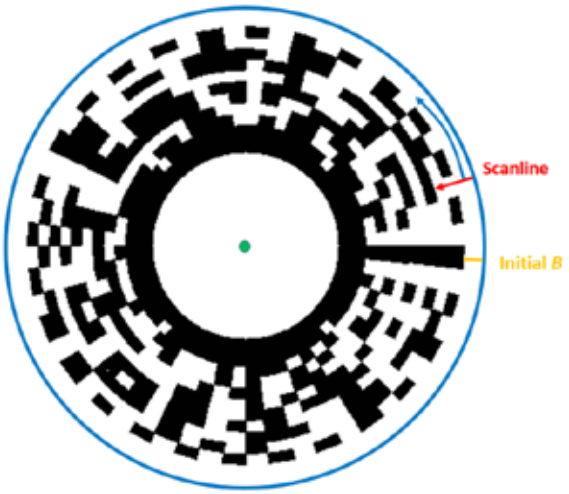

Figure 8: Scanline starting outside the barcode and revolving around its center

put image to the accurately perspective corrected image in one transformation step. This more accurately undistorted image is the input of the following last two stages.

Detection of Radial Timing Information

The radial zebra pattern defines how many rings a barcode has and how far apart they are spaced. It is always situated next to a start marker. Because the barcode has possibly been mirrored, it could lie either to the right or to the left of the start marker. Therefore, two radial lines, one on each side of a start marker, are projected from the barcode center through the potential locations of the vertical zebra pattern. To evaluate whether one of these lines goes through a radial zebra pattern, the alternating continuous stretches of black and white pixels are measured. Because all rings are equal in thickness, a perfect zebra pattern would have n continuous black pixels followed by n continuous white pixels followed by n continuous black pixels again. If no continuous stretch of equally colored pixels significantly deviates in length from the average, a radial zebra pattern has been found.

The radial zebra patterns also determine the angular reading direction of the barcode payload. If it is to the right of the start marker, the reading direction is clockwise.

Evaluation of the Modules

In the last stage of the decoder-chain the colors of all modules inside the data zone have to be evaluated and stored in the right order in a bit vector.

The information provided by the angular and the radial zebra patterns in combination with the barcode center allows us to virtually draw a grid over the barcode in the undistorted input image. Figure 9 depicts such a grid. At every intersection of the grid a module has to be evaluated as either black or white. To evaluate each module, a total of five pixels are sampled in a cross-shaped pattern. Each pixel votes for the module to be either black or white according to its color. Because the sampling is performed on a binarized image and an uneven number of pixels are sampled, a module's color is never ambiguous, so it can always be classified as either black or white.

Storage

The maximum amount of data our annular barcode design supports depends on the number of barcode modules, the datatype (input character set), and error correction level. All these parameters are equal to those in a QR codes. Therefore, the concepts and techniques used in QR codes are also applicable to our annular barcodes.

Tests

During the development phase, most tests were done using a laptop webcam as scanner to read a barcode shown on a mobile phone display. The scanning process is fast and reliable enough to cause no inconvenience to the user, even under non-ideal conditions such as a considerable amount of perspective distortion.

Because the results are dependent on the actions of the person holding the barcode, it is impossible to reliably quantify the success rate and overall quality with this test method. To measure the quality and capabilities of the decoder in a reproducible way, a batch testing tool has been developed. It generates a barcode, adds some defects to it and then checks if it can be correctly decoded. To obtain reproducible results, each test is run several times.

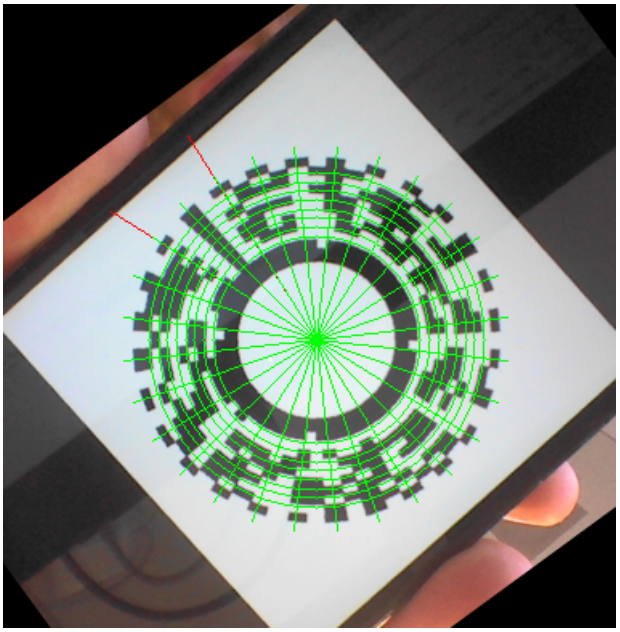

Figure 9: Grid matching the barcode modules. To keep the grid visualization readable a line was drawn only through black angular modules.

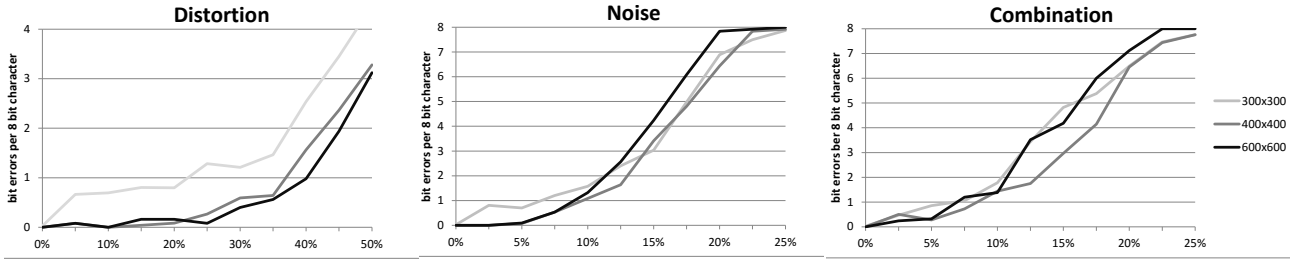

Figure 10: Test results: left) Distortion up to 35% only results in an average bit error larger than 1 in the smallest barcode; middle) Noise up to 10% results in an average bit error of less than 2; right) Distortion up to 10% and noise up to 10% together result in an average bit error of less than 2.

Out test tool is capable of adding two types of defects to a barcode:

- *Perspective distortion:* The perspective distortion can be controlled in percent. 0% means no distortion, with 100% it is possible that all four corners of the barcode image are mapped to a single line in the distorted image.
- *Salt and pepper noise:* This defect adds random black and white pixels to the image and is controlled by a percentage value. At 100%, the image is fully covered by noise.

All tests were conducted with three small barcodes of type 1 (see Table 1). The influence of the two different defect types has been tested both in individual test series and in a series of tests with both defects combined. Figure 10 shows the average bit errors depending of the amount of distortion and noise. These diagrams help to choose an appropriate error correction level. For example, if we use an error correction code which is able to correct two bit errors per eight bit character, then our annular barcodes can be distorted up to 35-40% (depending on barcode size) and contain noise up to 10%.

Conclusion and Further Work

We have presented a novel generic design for annular barcodes and we have shown that our decoder is able to decode very small barcodes (5 mm diameter at 300 dpi) of type 1 under real conditions with a small average bit error rate that can be corrected with an appropriate error correction code. On our test machine (i7-3770 quad core with HT @ 3.4 GHz, 12 GiByte RAM) the mean decoding times for the three barcode sizes are 13, 11, and 18 ms, respectively.

There is a chance that the data zone in a barcode forms a pattern similar to the start marker. If such a pattern is present, then it might be possible that the decoder selects the wrong one and hence the payload cannot be read. The chance of a pattern equal to the start marker occurring in a barcode is dependent on the number of rings and the number of modules per ring. Due to the zebra pattern around the outside of the barcode, a start marker lookalike can only occur at every second angular module. For a barcode that has eight rings, the chance of a lookalike is about 1:50000. However, if the number of rings is reduced this quickly becomes a big problem. A possible solution is to check the generated barcode for start marker lookalikes. If such a lookalike is detected, a random bitmask is selected and an XOR-Operation is performed on the data zone and the bitmask. To be able to decode the barcode again the used bitmask has to be encoded into the barcode data zone as additional header information.

So far we focused on very small angular barcodes and therefore on type 1 with just one start marker. Further work should be done for larger barcodes of type 2 and 3.

Size	Module Size	Min. Radius	Max. Radius
300x300	6	60	120
400x400	8	80	160
600x600	12	120	240

Table 1: Annular barcode parameters (pixels) used in our tests

References

[BC09] Barcoding Connected: Shotcode Barcode: The circular 2D barcode for mobile tagging: http://blog.barcoding.com/2009/02/shotcode-barcode-the-circular-2d-barcode-for-mobile-tagging/

[Brü15] Brügger, Robin. Ring-shaped Barcodes. Bachelor Thesis, Computer Science, University of Applied Sciences Northwestern Switzerland, FHNW, 2015.

[Mal15] Mallick, Satya. Blob Detection Using OpenCV (Python, C++). http://www.learnopencv.com/blob-detection-using-opencv-python-c/

[Mud06] Mudie, Stuart: Shotcode. Licensed under CC BY-SA 2.0 via Wikimedia Commons: https://commons.wikimedia.org/wiki/File:Shotcode.png

[TR] TechnoRiver. Barcode software and components: http://www.technoriversoft.com/CircularBarcode.html

[Wiki] Barcode: https://en.wikipedia.org/wiki/Barcode

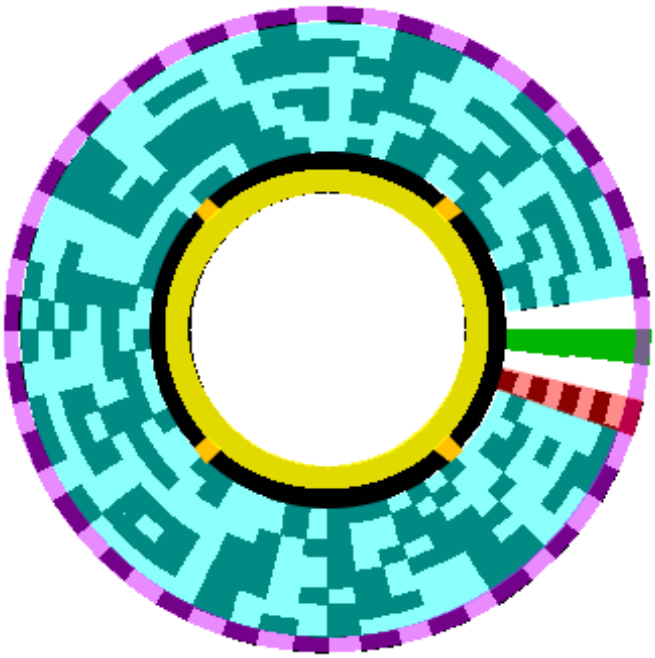

Figure 4: Annular barcode of type 1 with a data zone of 56 bytes with highlighted special purpose zones: a yellow ring for locating the barcode, orange square markers for rough perspective correction, a blue data zone containing all data modules, a green start marker defines the beginning of the data zone, a red angular zebra pattern determines the angles between the modules, and a purple radial zebra pattern determines the radial timing of the rings.

Fachhochschule Nordwestschweiz
Institut für Mobile und Verteilte Systeme
Bahnhofstrasse 6
CH-5210 Windisch

www.fhnw.ch/technik/imvs
Tel. +41 56 202 99 33