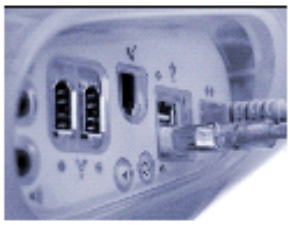
Technical Reports 1

Zentrum für Informationstechnik-Gestaltung
ITeG


Axel Hoffmann, Stefan Niemczyk (Editoren)

Die VENUS - Entwicklungsmethode

Eine interdisziplinäre Methode für soziotechnische Softwaregestaltung

kassel university press

ITeG Technical Reports

Band 1

Herausgegeben vom
Zentrum für Informationstechnik-Gestaltung (ITeG)
an der Universität Kassel

Zentrum für Informationstechnik-Gestaltung

Universität Kassel
ITeG Zentrum für
Informationstechnik-Gestaltung
Pfannkuchstraße 1
D-34121 Kassel

Die VENUS-Entwicklungsmethode

Eine interdisziplinäre Methode für soziotechnische Softwaregestaltung

Axel Hoffmann, Stefan Niemczyk (Editoren)

kassel university press

Projektleiter:

Prof. Dr.-Ing. Klaus David
FG Kommunikationstechnik

Prof. Dr. Jan Marco Leimeister
FG Wirtschaftsinformatik

Prof. Dr.-Ing. Ludger Schmidt
FG Mensch-Maschine-Systemtechnik

Prof. Dr. Kurt Geihs (Sprecher)
FG Verteilte Systeme

Prof. Dr. Alexander Roßnagel
FG Öffentliches Recht

Prof. Dr. Gerd Stumme
FG Wissensverarbeitung

Prof. Dr. Arno Wacker
FG Angewandte Informationssicherheit

Autoren:

Martin Atzmüller
Harun Baraki
Kay Behrenbruch
Diana Comes
Christoph Evers
Axel Hoffmann
Holger Hoffmann
Silke Jandt
Mark Kibanov
Olga Kieselmann
Romy Kniewel
Immanuel König
Björn-Elmar Macek
Stefan Niemczyk
Christoph Scholz
Michaela Schuldt
Thomas Schulz
Hendrik Skistims
Matthias Söllner
Christian Voigtmann
Andreas Witsch
Julia Zirfas

Gestaltung technisch-sozialer Vernetzung in situativen ubiquitären Systemen (VENUS)

Gefördert durch:

LOEWE
Exzellente Forschung für Hessens Zukunft

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.dnb.de abrufbar

ISBN: 978-3-86219-550-3
URN: URN: http://nbn-resolving.de/urn:nbn:de:0002-30870

Vorwort der Herausgeber

Dies ist der erste Band einer neuen Serie: „ITeG Technical Reports“. In loser Reihe werden hier aus dem Zentrum für Informationstechnik-Gestaltung (ITeG) Forschungs- und Entwicklungsergebnisse berichtet und der wissenschaftlichen Öffentlichkeit zur Diskussion gestellt. Das Zentrum für Informationstechnik-Gestaltung der Universität Kassel lässt sich in seinem Forschungsspektrum von der Idee leiten, die Informationstechnik der Zukunft unter Beachtung ihrer sozialen Einbettung gesellschaftlich wünschenswert zu gestalten. So wie Informationstechnik zu einem strukturbildenden Element für unsere heutige „digitale Gesellschaft“ wird, sehen wir auch die wachsende Notwendigkeit einer interdisziplinären Gesamtsicht auf Mensch und Technik. Die Gestaltung von Informationstechnik wird somit zur soziotechnischen Herausforderung. Das ITeG liefert methodische Unterstützung für diese Gestaltungsaufgabe.

Ein Kernprojekt des ITeG und im obigen Sinne identitätsstiftend war der Forschungsschwerpunkt „VENUS: Gestaltung technisch-sozialer Vernetzung in situativen ubiquitären Systemen“, welcher durch die 2. Förderstaffel der hessischen Landes-Offensive zur Entwicklung Wissenschaftlich-ökonomischer Exzellenz (LOEWE) von 2010-2013 gefördert wurde. Die besondere Herausforderung hierbei war, bereits bei der Entwicklung der Technik neben den funktionalen Aspekten konsequent auch nicht-funktionale Aspekte, wie beispielsweise das Recht auf informationelle Selbstbestimmung und Fragen der Benutzbarkeit, in das Design einzubeziehen. Dies erforderte eine disziplinübergreifende Zusammenarbeit. Trotz aller Bekenntnisse zur interdisziplinären Kooperation in der Technikgestaltung sind die Probleme der disziplineigenen Begriffswelten und Sprachen, der Heterogenität der Anforderungen aus verschiedenen Disziplinen, der zeitlichen Synchronisation der domänenspezifischen Beiträge, fehlender Brückenkonzepte und mangelnder gemeinsamer Theorien und Methoden im Grunde noch weitgehend ungelöst. Genau diese Probleme wurden im VENUS-Forschungsschwerpunkt an der Universität Kassel aufgegriffen und für die Gestaltung insbesondere von Anwendungen des Ubiquitous Computing systematische Lösungen erarbeitet und Möglichkeiten für weitergehende Verallgemeinerungen deutlich gemacht.

Das multi-disziplinär besetzte VENUS-Forscherteam mit Experten aus den Bereichen Informatik, Mensch-Maschine-Schnittstelle, Vertrauensmanagement und Recht hat in einem iterativen Entwicklungsprozess eine systematische interdisziplinäre Entwicklungsmethode für die Gestaltung von ubiquitären Systemen erarbeitet. Der vorliegende Technische Bericht beschreibt die dabei entstandene VENUS-Entwicklungsmethode über den kompletten Entwicklungszyklus von der Bedarfsanalyse bis hin zur Systemevaluation. Die einzelnen Phasen der Softwareentwicklung werden durch integrierte Methoden und Techniken zur iterativen interdisziplinären Entwicklung sozialverträglicher ubiquitärer Systeme unterstützt. Es herrscht mittlerweile weitgehende Einigkeit darüber, dass solcher Art soziotechnische Systementwicklung zu Systemen führt, die von den Endnutzern mehr akzeptiert werden und einen größeren Mehrwert für alle Beteiligten bieten. Die hier vorgeschlagenen methodischen Bausteine sind für die Umsetzung eines soziotechnischen Ansatzes in der Praxis als Erweiterung existierender Softwareentwicklungsmethoden gedacht.

Alle am VENUS-Projekt Beteiligten würden sich über Kommentare und Erfahrungsberichte aus der Praxis zur VENUS-Entwicklungsmethode freuen.

Prof. Dr. Kurt Geihs

Inhalt

1 Einleitung

Das Paradigma des Ubiquitous Computing (UC) ermöglicht neue Formen der Informationsverarbeitung. Ubiquitäre Systeme ermitteln ihren aktuellen Nutzungskontext mit Hilfe verschiedener Sensoren und passen so ihre Dienste an die jeweilige Situation an. Ziel ist es, dem Nutzer den bestmöglichen Service zu bieten. Dazu kooperieren sie mit weiteren ubiquitären Systemen in ihrer Umgebung. Bei der Entwicklung solcher Systeme ergeben sich neue Herausforderungen, bedingt durch technologische Aspekte wie auch durch die starke Integration des Nutzers und seines sozialen Kontextes. Die Bedienbarkeit des Systems, das Vertrauen der Nutzer in die Technik und die Einhaltung von Gesetzen zum Schutz personenbezogener Daten sind wichtige Aspekte solcher Systeme. Um eine hohe Nutzerakzeptanz zu erreichen, ist es notwendig, ubiquitäre Systeme bestmöglich in die Handlungen der Nutzer zu integrieren. So können die Systeme die Nutzer bestmöglich beim Erreichen ihrer Ziele unterstützen. Darüber hinaus trägt eine Integration des Systems in das Umfeld der Nutzer, und somit die Bereitstellung der Dienstleistung zu jeder Zeit und an jedem Ort, weiter zur Akzeptanz bei. Die erfolgreiche Integration eines ubiquitären Systems ist einer der Schlüsselfaktoren zum Erfolg des Systems.

Für die erfolgreiche Einbeziehung des Faktors Mensch in die Entwicklung moderner Software mangelt es momentan noch an systematischen Entwicklungsmethoden und Werkzeugen. Bernstein et al. [1] argumentieren, dass die meisten Anwendungen mit einem trivialen “trial-and-error” Ansatz entwickelt werden, was häufig dazu führt, dass Entwicklungsprojekte nicht erfolgreich sind. Immer mehr setzt sich daher die Erkenntnis durch, dass zukünftig Entwickler nicht nur die klassische Rolle des Softwarearchitekten einnehmen, sondern ebenfalls organisationsspezifische und soziale Aspekte bei der Systementwicklung beachten müssen. In einem ähnlichen Kontext wurde bereits 1951 von Trist und Bamforth [2] ein entsprechendes Paradigma vorgestellt. Viele traditionelle soziotechnische Perspektiven hingegen, z. B. [3], [4], [5], beschränken die technische Perspektive auf ein Minimum. Eine Ausnahme bilden Baxter und Sommerville [6], [7] mit ihrem Ansatz, welcher die Bedeutung der sozialen Aspekte in der Systementwicklung neben den technischen Aspekten betont.

Das Ziel des Forschungsschwerpunktes VENUS (Gestaltung technisch-sozialer Vernetzung in situativen ubiquitären Systemen), gefördert durch die 2. Förderstaffel der hessischen Landes-Offensive zur Entwicklung Wissenschaftlich-ökonomischer Exzellenz (LOEWE), war die Entwicklung einer systematischen und interdisziplinären Entwicklungsmethode, welche die funktionalen und nicht-funktionalen Designaspekte von UC berücksichtigt. Die Entwicklung wurde durch ein interdisziplinäres Forschungsteam mit Experten aus den Bereichen Informatik, Gebrauchstauglichkeit, Vertrauensmanagement und Recht durchgeführt. Dieser Bericht beschreibt die dabei entstandene Entwicklungsmethode. Dafür werden verschiedene Phasen der Softwareentwicklung aufgegriffen, in denen Methoden und Techniken die interdisziplinäre Entwicklung sozialverträglicher ubiquitärer Systeme unterstützen können. Diese Methoden sind als Erweiterung für bereits bestehende Entwicklungsmethoden gedacht, so dass sie eine Ergänzung zur bestehenden Praxis darstellen.

1.1 Ubiquitäre Systeme

Der Begriff Ubiquitous Computing (UC) wurde von Mark Weiser geprägt. Weiser charakterisierte UC als eine Zukunftstechnologie, die sich in unsere physikalische Umwelt einbindet, mit ihr verschmilzt und eine komplexe Unterstützung durch Informationstechnologie ermöglicht, ohne dass die "Computer" wahrgenommen werden und eine explizite Bedienung der Rechner stattfindet [8], [9]. Ubiquitäre Systeme lassen sich durch drei Besonderheiten definieren:

- Erstens zeichnen sich UC-Systeme durch eine Einbettung der Technik in die alltägliche Lebensumgebung der Nutzer aus, was stets mit dem automatischen Erfassen und Verarbeiten von Daten über den Nutzer und den Nutzungskontext verbunden ist.

- Zweitens sind ubiquitäre Systeme kontextsensitiv und adaptiv. Der Nutzer wird passend zu seinem aktuellen Kontext unterstützt. Dabei adaptieren sich die ubiquitären Dienste in Echtzeit an sich wandelnde Kontexte und binden sich in verschiedene technische Infrastrukturen ein.

- Drittens sind ubiquitäre Systeme ortsübergreifend und vielfältig vernetzt. Die Unterstützung durch ubiquitäre Systeme ist für den Nutzer quasi „überall“ und „jederzeit“ über unterschiedliche Schnittstellen zugänglich.

Der Nutzer interagiert mit UC nicht durch ein explizites "Bedienen" des Systems. Vielmehr erfolgt die Nutzung implizit und parallel zur Alltagshandlung und verändert diese nicht wesentlich. Die Unterstützung geschieht gleichzeitig zu großen Teilen außerhalb oder in der Peripherie der Aufmerksamkeit des Nutzers. UC-Systeme treffen auf der Grundlage der erfassten Daten autonom Entscheidungen und sind infolgedessen der Kontrolle der Nutzer in höherem Maße entzogen, als dies bei anderen softwarebasierten Systemen der Fall ist. Die Herausforderungen für die Entwickler von UC-Systemen liegen darin, das Informations- und Kontrollbedürfnis der Nutzer ausreichend zu berücksichtigen, ohne dabei die bessere Gebrauchstauglichkeit, bezogen auf Effizienz und Effektivität, aufzugeben. Eine wichtige Rolle spielt in diesem Zusammenhang das Vertrauen der Nutzer in die Unterstützungsleistung des Systems. Eine weitere Herausforderung bei der Gestaltung von UC-Systemen liegt darin, die vorhandenen Datenströme einerseits ubiquitär einbinden zu können und gleichzeitig die anfallenden personenbezogenen Daten gegen den unbefugten Datenzugriff zu sichern. Die Akzeptanz beim Nutzer hinsichtlich einer UC-Anwendung hängt in direkter Weise von dem Grad der Erfüllung dieser soziotechnischen Anforderungen ab.

1.2 Entstehungsgeschichte

Eine sozialverträgliche Technikgestaltung ermöglicht es, informationstechnische Innovationen so zu gestalten, dass sie nachhaltig nutzbar und nutzenstiftend sind. Die technische Machbarkeit und Umsetzung technischer Innovationen alleine reichen für den Erfolg häufig nicht aus. Die Einbeziehung der soziotechnischen Perspektive ist hier gefragt. Wichtige Anforderungen ergeben sich beispielsweise aus den Bereichen Ergonomie, Ökonomie und Recht. Zentrale Herausforderung für die Realisierung des Innovationspotentials ist die erfolgreiche Einbettung allgegenwärtiger Informationstechnologie in individuelles, gemeinschaftliches, gesellschaftliches und wirtschaftliches Handeln.

Für eine gelungene Einbettung sind im Projekt VENUS exemplarisch die Erfolgsfaktoren Gebrauchstauglichkeit der Systeme, das Vertrauen der Nutzer in die Technik, die Erfüllung der rechtlichen Rahmenbedingungen sowie ökonomisch attraktive und nachhaltige Geschäfts-, Service- und Betreibermodelle von Interesse. Um diese Erfolgsfaktoren bereits bei der Entwicklung der Technik zu berücksichtigen, ist eine systematische Entwicklungsmethode erforderlich. Normative Anforderungen, die beschreiben, welche Bedingungen eine Technikinnovation systematisch erfüllen muss, um langfristig eine akzeptierte und akzeptable Einbettung zu erreichen, werden auf einer abstrakten Ebene als Anforderungen der Sozialverträglichkeit zusammengefasst. Diese abstrakten, meist „nicht messbaren“ Anforderungen der Sozialverträglichkeit werden in konkrete technische Artefakte überführt.

Um die Methode im Sinne der Anforderungen und Ziele zu entwickeln, wurde ein iteratives Verfahren gewählt. Startpunkt waren Workshops, in denen Informatiker zusammen mit Experten aus den Bereichen Gebrauchstauglichkeit, Vertrauenswürdigkeit und Rechtsverträglichkeit die Anforderungen an eine interdisziplinäre Methode analysierten und ein initiales Konzept der Methode entwickelten [10]. Dieses Konzept wurde daraufhin innerhalb eines Projektes zur Entwicklung eines soziotechnischen Systems angepasst. Hierbei war es besonders wichtig, vor allem die Voraussetzungen und Ergebnisse der einzelnen Fachexpertisen aufeinander abzustimmen, so dass die konstruktive Zusammenführung der einzelnen Aktivitäten gegeben war. Das nun bestehende Konzept der iterativen Methode ist in einem Arbeitspapier festgehalten, welches die Grundlage für diesen technischen Bericht bildete.

Bei der iterativen Weiterentwicklung der Methode wurde diese nacheinander in verschiedenen Entwicklungsprojekten eingesetzt, evaluiert und anhand der Ergebnisse angepasst (siehe [11], [12], [13]). Die Projektteams bestanden jeweils aus sechs bis acht Entwicklern und Wissenschaftlern. Dabei fanden ubiquitäre Anwendungen in den Anwendungsdomänen mobile soziale Vernetzung im privaten Bereich, selbstbestimmtes Leben im Alter sowie soziale Vernetzung am Arbeitsplatz Eingang in die Untersuchungen. Im Anschluss an jedes Entwicklungsprojekt wurde die Methode auf Basis der Erfahrungen der Beteiligten überarbeitet und weiterentwickelt.

1.3 Übersicht

Die VENUS-Entwicklungsmethode beschreibt einen iterativen Entwicklungsprozess und besteht aus den Phasen Bedarfsanalyse, Anforderungsmanagement, Konzeptdesign, Softwaredesign / Implementierung und Evaluation in der Nutzung (Abbildung 1). Zusätzlich zu einer abschließenden Evaluation in der Nutzung wird Evaluation auch begleitend zur Bedarfsanalyse (Evaluation von Anwendungsszenarien) und zur Softwaredesign/Implementierung (Evaluation von Teilfunktionen) eingesetzt. Ein solches Vorgehen legt besonderen Wert auf die laufende Rückkopplung von Evaluationsergebnissen in den Entwicklungsprozess und liefert die Kriterien für die angestrebte Iterationssteuerung. In den Phasen des Anforderungsmanagements wird neben den üblichen funktionalen und nichtfunktionalen Anforderungen der informationstechnischen Seite vor allem auf nichtfunktionale Anforderungen der Bereiche Recht, Vertrauen und Gebrauchstauglichkeit Wert gelegt. Ziel der Methodik ist die systematische Gestaltung der Systeme.

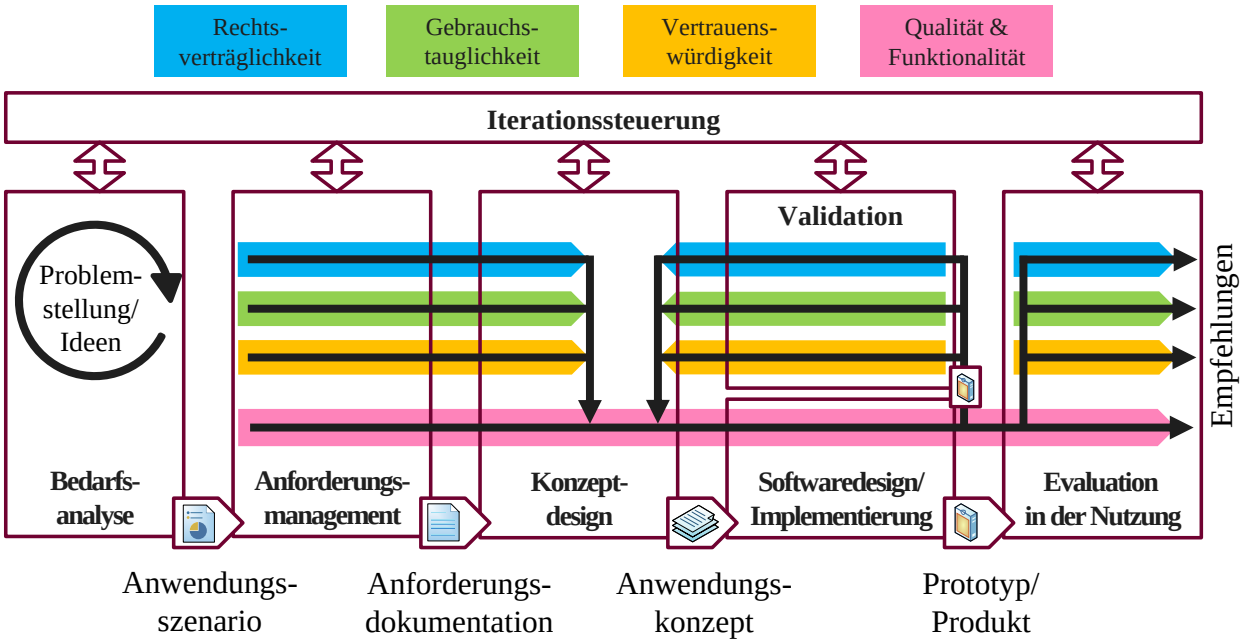


Abbildung 1: Überblick über die VENUS-Entwicklungsmethode

Um die Merkmale Rechtsverträglichkeit, Gebrauchstauglichkeit, Vertrauenswürdigkeit und Softwarequalität in der Phase des Anforderungsmanagements zu koordinieren, ist es wichtig, bereits im Vorfeld ein gemeinsames Verständnis von der Anwendung (dem Produkt) zu entwickeln. Diese gemeinsame Version wird in der Bedarfsanalyse in Form eines Anwendungsszenarios erstellt. Um ubiquitäre Anwendungsszenarien zu entwerfen, ist es notwendig, dass sich die Beteiligten von ihren Vorstellungen über klassische Computersysteme lösen. So entstehen in der Bedarfsanalyse ein oder mehrere Anwendungsszenarien. Um das Anwendungsszenario noch deutlicher zu machen, werden unter anderem Personas, das sind archetypische Beschreibungen von fiktiven Nutzergruppen, eingesetzt. Das Anwendungsszenario stellt sicher, dass vor der Anforderungserhebung ein gemeinsames Verständnis über die Anwendung existiert und die Anforderungen der Merkmale Rechtsverträglichkeit, Gebrauchstauglichkeit, Vertrauenswürdigkeit und Softwarequalität zielgerichtet erhoben werden können.

Im weiteren Konstruktionsprozess werden die Szenarien von den Entwicklern als Referenz während der technischen Entwicklungsarbeit genutzt, um die aus Nutzersicht gesteckten Ziele nicht aus den Augen zu verlieren. Zudem dienen die Szenarien bei der Evaluation von Teilfunktionalitäten parallel zur Implementierung als Grundlage für die Beurteilung durch Experten.

Rechtsverträglichkeit

Für den Bereich der rechtlichen Anforderungen und auch als Ausgangspunkt für die VENUS-Entwicklungsmethode insgesamt wird der KORA-Ansatz verwendet. KORA steht für Konkretisierung rechtlicher Anforderungen. Die KORA-Methode gewinnt technische Anforderungen aus rechtlichen Vorgaben, die sich aus den grundlegenden generell-abstrakten Rechtsnormen auf den oberen Ebenen der Rechtshierarchie – insbesondere der Verfassung – ergeben. Dabei geht die Rechtsverträglichkeit über das Konzept der Rechtmäßigkeit hinaus, indem sie nicht nur eine minimale, sondern eine optimale Umsetzung rechtlicher Anforderungen anstrebt.

Gebrauchstauglichkeit

Im Fokus der Ergonomie steht die Gebrauchstauglichkeit der zu entwickelnden UC-Systeme. Die Gebrauchstauglichkeit ist gegeben, wenn die Nutzer Ihre Ziele mit dem System auf effiziente und zufriedenstellende Art und Weise erreichen. Für UC steht diesem zielorientierten Gedanken die Frage gegenüber, ob die Nutzer die UC-spezifischen Besonderheiten auch akzeptieren. Das ergonomische Thema der Nutzerakzeptanz steht in einem engen Zusammenhang mit Fragen der Rechtsverträglichkeit und der Vertrauenswürdigkeit.

Vertrauenswürdigkeit

Für den Bereich Vertrauen integriert die VENUS-Entwicklungsmethode das Vorgehen des Trust Engineering. Trust Engineering nutzt theoretische Grundlagen von Vertrauen, um vertrauensunterstützende Komponenten einer Anwendung zu entwerfen. Dabei adressiert die Methode die Determinanten von Vertrauen, die durch spezielle Designentscheidungen beeinflusst werden können. Mit Hilfe des Szenarios werden im Interaktionsprozess des fiktiven Nutzers mit der Anwendung Situationen identifiziert, in denen er mit verschiedenen Unsicherheiten konfrontiert ist. Die Unsicherheiten werden nach Relevanz priorisiert. Für die Unsicherheiten werden Determinanten von Vertrauen herausgesucht, mit denen diesen begegnet werden kann. Hier werden Anforderungen formuliert, welche die Erfüllung der vertrauensfördernden Determinanten verlangen.

Softwarequalität

Anforderungen der Stakeholder an Funktionalität und Qualität, die Grundlage eines jedes Entwicklungsprojektes sind, werden mit den vorher genannten Anforderungen zusammengeführt. Zusätzlich werden speziell Anforderungen der Softwarequalität aus relevanten Normen abgeleitet, die frühzeitig bei der Konzeption der Systeme Berücksichtigung finden.

Die Anforderungen aller Merkmale (Rechtsverträglichkeit, Gebrauchstauglichkeit, Vertrauenswürdigkeit, Softwarequalität) werden im nächsten Schritt vereinbart. Dies geschieht nach dem EasyWinWin Vorgehen, ein werkzeuggestütztes Verfahren zur systematischen Erhebung und Verhandlung von Anforderungen in Software-Projekten. Als Ergebnis entsteht eine Anforderungsdokumentation, die als Artefakt in die nächste Phase überführt wird.

Im Konzeptdesign werden die Anforderungen in ein Anwendungskonzept überführt. Dieses Konzept besteht aus UseCases (Anwendungsfälle), also die Bündelung aller möglichen Szenarien die eintreten können, wenn der Nutzer versucht, mit Hilfe des Systems sein Ziel zu erreichen, Flowcharts, das sind Ablaufdiagramme der Funktionsweise, der Definition der Benutzungsschnittstelle sowie den Schnittstellen und Datenflüssen.

Das Anwendungskonzept wird dann in einem iterativen Prozess implementiert. Dabei werden die entstehenden Prototypen durch Experten darauf überprüft, ob die vorher definierten Anforderungen bei der Umsetzung berücksichtigt wurden. Durch diese Validierung können Änderungen am Anwendungskonzept durchgeführt werden, die in die nächste Iteration eingehen. Zudem werden die im Prozess entstehenden Teilfunktionen aus Nutzersicht evaluiert. Die Evaluation von Teilfunktionalitäten und eine direkte Rückkopplung der Ergebnisse in das Softwaredesign erfordert eine enge Abstimmung zwischen den Programmierern und den Usability-Experten. Agile Methoden der Softwareentwicklung (z. B. Extreme Programming oder SCRUM) bieten in Form der regelmäßig stattfindenden Besprechungen aller Stakeholder eine gute Voraussetzung für die notwendige Kommunikation. Zudem ist die agile Softwareentwicklung darauf ausgerichtet, konstant lauffähige und testbare Software vorzuhalten und in regelmäßigen Abständen lauffähige Entwicklungszustände zu liefern, die evaluierbar sind. Die VENUS-Entwicklungsmethode bietet hier für verschiedene bekannte Vorgehensmodelle der Softwareentwicklung Konnektoren, die ein einfaches Miteinander erlauben. Somit sind die hier vorgestellten Methoden und Techniken eine Ergänzung zu vorhandenen Vorgehen.

Am Ende des Entwicklungsprozesses steht die Evaluation in der Nutzung durch reale Versuchsnutzer. Hiermit soll sichergestellt werden, dass sich das entwickelte technische System im tatsächlichen soziotechnischen Kontext so verhält, wie es zuvor angedacht war. Dabei können durch die Evaluation in den Expertenbereichen Empfehlungen für weitere Projektaktivitäten generiert werden.

1.4 Aufbau und Rollen

Die wesentlichen Inhalte der VENUS-Entwicklungsmethode sind in den modularen, aufeinander aufbauenden Vorgehensbausteinen (VB) enthalten. Jeder Vorgehensbaustein ist eine eigenständige Einheit und einzeln änder- bzw. erweiterbar. Ein Vorgehensbaustein beinhaltet alle Bestandteile, die zur Bearbeitung einer konkreten Aufgabenstellung im Laufe eines Entwicklungsprojektes notwendig sind. Ein Vorgehensbaustein kapselt dabei diejenigen „Produkte“ und „Aktivitäten“, die für die Erfüllung dieser Aufgabenstellung relevant sind und damit inhaltlich zusammengehören, wie beispielsweise die Inhalte des Ideen-Generierens oder der Erarbeitung von Anwendungszielen. Bei jedem Vorgehensbaustein stehen ein oder mehrere Aktivitäten im Vordergrund. Das Ergebnis eines Vorgehensbausteins ist ein Produkt. Zudem werden Beispiele zur Veranschaulichung angeführt. Als durchgängiges Beispiel bei der Beschreibung der Vorgehensbausteine wird die im Rahmen von VENUS entwickelte Demonstratoranwendung Meet-U gewählt, welche im nächsten Abschnitt vorgestellt wird.

Für einen Vorgehensbaustein gibt es jeweils eine „verantwortliche Rolle“ und gegebenenfalls mehrere „mitwirkende Rollen“. Dabei können mehrere Rollen durchaus von ein und derselben Person eingenommen werden. Bei der VENUS-Entwicklungsmethode kommen folgende Rollen zum Einsatz:

Projektleiter	Koordinator des gesamten Entwicklungsprojektes; plant und steuert die Durchführung der Vorgehensbausteine der VENUS-Entwicklungsmethode.
Anforderungsanalyst	Verantwortlich für die Konsolidierung der Anforderungen; hat Erfahrung mit der Fachsprache verschiedener Experten, der Sprache der Nutzer und des Entwicklungsteams.
System Engineer	Verantwortlicher Leiter der Systementwicklung und Implementierung; wendet das gewählte Vorgehensmodell bei der Umsetzung an.
Software Engineer	Experte für Softwareentwicklung; gestaltet und implementiert Software.
Jurist	Experte für das Thema IT-Recht; hat weitreichendes Wissen auf dem Gebiet des Datenschutzes und weiterer technisch relevanter Gesetze und Bestimmungen.
Trust Engineer	Experte für die Vertrauenswürdigkeit und Akzeptanz von Anwendungssystemen; kennt Risikofaktoren, die Vertrauen bedingen, und die Determinanten zur Vertrauenswürdigkeit von Systemen.
Usability Engineer	Experte auf dem Gebiet der Gebrauchstauglichkeit; kennt relevante Normen sowie Best Practices und kann Methoden des Usability Engineering anwenden.
IT-Sicherheitsexperte	Experte für technische Sicherheit; kennt Standardverfahren zur Überprüfung der IT-Sicherheit und steuert diese zum Entwicklungsprozess bei.

1.5 Die Anwendung Meet-U

Meet-U ist eine kontextsensitive, adaptive, mobile Anwendung, die auf der Android-Plattform für Smartphones basiert. Meet-U unterstützt die soziale Vernetzung. Ziel ist es, den Nutzer bei der Planung eines gemeinsamen Treffens mit Freunden und Bekannten bei einer Veranstaltung (Event) zu unterstützen. Die Anwendung unterstützt den Nutzer in verschiedenen Situationen: beim Planen von Events, auf dem Weg zum Event und auch beim Event vor Ort. Die Veranstaltung kann entweder privat (für den engen Freundeskreis) oder öffentlich (für alle) sein. Falls eine Veranstaltung von öffentlichen Organisationen angeboten wird, wie z. B. ein Konzert oder eine Sportveranstaltung, gilt sie als öffentlich und ist von allen Nutzern einsehbar. Es wird angenommen, dass Informationen über öffentliche Veranstaltungen mittels externer Web-Dienste verfügbar sind. Meet-U kann selbstständig und dynamisch zur Laufzeit externe Dienste aus dem Internet oder einem lokal verfügbaren Netz (z. B. WLAN) entdecken und einbinden. Diese können von verschiedenen Dienstanbietern stammen, um dem Nutzer zusätzliche Informationen und Funktionalitäten anzubieten. Die Dienste können ebenfalls dazu genutzt werden, die Adaptionsentscheidung (d. h. wann macht die Anwendung was) zu beeinflussen, indem zusätzliche Kontextinformationen bereitgestellt werden.

Bevor der Nutzer die Meet-U-Anwendung verwenden kann, muss er sich mit Nutzername und Passwort anmelden. Hat er noch kein Konto, so kann er sich direkt am Gerät registrieren. Dabei gibt er sein Nutzerprofil an, welches Name, Vorname und Präferenzen (z. B. Vorliebe für Kino oder Sportveranstaltungen) beinhaltet. Meet-U schlägt dem Nutzer, unter Berücksichtigung seiner Präferenzen, für ihn interessante Veranstaltungen vor. Im anschließenden Planungs-Modus kann der Nutzer seinen Freunden Einladungen für die Teilnahme an einer Veranstaltung schicken. Auf dem Weg zur Veranstaltung (im so genannten Navigations-Modus) erhält der Nutzer Navigationsinformationen, die ihm beim Finden des Veranstaltungsortes helfen sollen. Ist der Nutzer am Veranstaltungsort angekommen, können weitere lokale Dienste in Meet-U integriert werden, um beispielsweise den Nutzer bei der Navigation im Veranstaltungsgebäude zu unterstützen. Dadurch verfügt er über detaillierte Informationen, um den genauen Veranstaltungsraum oder seinen Zuschauerplatz zu erreichen. Auch andere vor Ort verfügbare Event-Dienste können von Meet-U eingebunden werden, um dem Nutzer zusätzliche Informationen und Funktionalitäten für die Veranstaltung (z. B. Kartenkauf, Programminformationen, Bilderaustausch) anzubieten.

Im Folgenden verwenden wir Meet-U als durchgängiges Beispiel bei der Beschreibung der Vorgehensbausteine der VENUS-Entwicklungsmethode. Diese bestehen zum einen aus bekannten Bausteinen (wie der Ideenfindung mittels Brainstorming) und auch neuen Bausteinen (wie dem Anforderungsmanagement).

2 Bedarfsanalyse (Phase 1)

Besonderheiten der Bedarfsanalyse in der VENUS-Entwicklungsmethode:

- Stakeholder verschiedenster Disziplinen werden von Anfang an einbezogen
- Stakeholdern werden die vielfältigen Möglichkeiten ubiquitärer Systeme verdeutlicht, um ihre Kreativität zu stimulieren
- Ein Geschäftsmodell für das ubiquitäre System wird von Anfang an mit entwickelt
- Ein abgestimmtes und verständliches Anwendungsszenario bildet die Grundlage für die nachgelagerten Entwicklungsphasen

Die Bedarfsanalyse besteht aus fünf Vorgehensbausteinen (VB). Sie dienen dem Zweck, Ideen zu konkretisieren, Probleme zu präzisieren und somit den Grundstein für Innovationen zu schaffen. Damit eine ubiquitäre Anwendung sinnvoll entwickelt werden kann, ist es essentiell, den Nutzungskontext zu verstehen und zu spezifizieren. Eine konkrete Problemstellung dient zum einen als Input für die Ideengenerierung und zum anderen als Input für die Erarbeitung der Anwendungsziele. Anwendungsziele dienen der Konkretisierung einer Anwendungsidee mit Bezug auf das zu entwickelnde System. Hierbei werden die Ziele gesammelt, die das System erfüllen sollte. Mit Hilfe der Anwendungsziele wird gezeigt, wie das System die Anwendungsidee operationalisieren wird. Mit der Hilfe von Personas werden aus den Zielen Anwendungsszenarios erstellt. Diese bilden die Grundlage für die weitere Entwicklung. Die fünf VB werden im Folgenden detailliert ausgeführt.

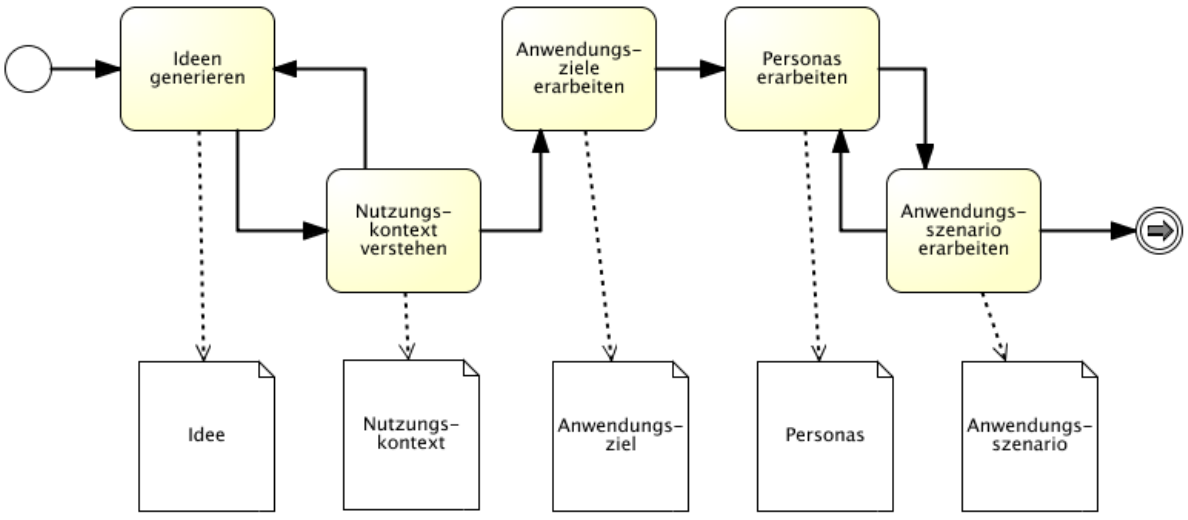


Abbildung 2: Vorgehensbausteine der Bedarfsanalyse

2.1 VB: Idee generieren

Besonderheiten der Ideengenerierung in der VENUS-Entwicklungsmethode:

- Stimuli für (zukünftige) ubiquitäre Systeme
- Erweiterung der mentalen Modelle der Stakeholder
- Erhöhung der Anzahl der Ideen durch den Einsatz von Group Support Systems (GSS)

Produkt	Idee
Verantwortliche Rolle	Projektleiter
Mitwirkende Rollen	Diverse

Das Ideengenerieren basiert auf dem Input einer konkreten, entweder neuen oder bestehenden Problemstellung. Wichtig ist es, das Problem und dessen Ursachen zu kennen (Siehe Teilaktivität Nutzungskontext verstehen). Bei der Ideengenerierung (bzw. der Generierung von Alternativen) sind insbesondere etablierte Kreativitätstechniken hilfreich und zielführend. Kreativitätstechniken dienen zur Sammlung von Ideen für Anwendungen. Unterstützend ist dabei, ein Modell der Zukunft zu erzeugen, um die Kreativität der Teilnehmer zu erhöhen. Eine etablierte und beliebte Kreativitätstechnik ist das Brainstorming. Je nach Fokus können auch andere Kreativitätstechniken zur Ideenfindung und -generierung angewandt werden [14]. In der Ideenfindung wird eine hohe Anzahl an Ideen generiert. Die Ideen werden anschließend priorisiert. Beider Schritte müssen getrennt voneinander ablaufen, da ansonsten die Priorisierung die Kreativität der Teilnehmer negativ beeinflussen kann.

2.1.1 Produkt[1]: Idee

Die Beschreibung einer konkreten Idee muss die folgenden strukturellen Mindestanforderungen erfüllen:

- Was ist es?
- Was tut es?
- Was ist der Nutzen?

Es können nach [15] fünf Qualitätsstufen einer Idee unterschieden werden:

- Die Idee ist logisch nachvollziehbar, in sich stimmig und kommunizierbar.
- Die Idee ist „reif“ bzw. komplett und damit bewertbar.
- Die Idee kann vom Unternehmen sinnvoll umgesetzt werden.
- Die Idee bietet dem Unternehmen wirtschaftliche Vorteile.
- Die Idee bietet der Gesellschaft Vorteile.

[1] Mit "Produkt" ist hier immer das Ergebnis eines Vorgehensbausteins gemeint (siehe Erläuterung 1.4, erster Absatz).

2.1.2 Aktivität: Ideen sammeln

Ideen können durch Hilfe von Kreativitätstechniken in Workshops entwickelt werden. Um ubiquitäre Anwendungsszenarien zu entwerfen, ist es notwendig, dass sich die Beteiligten von ihren Vorstellungen über klassische Computersysteme lösen. Dies kann durch den Einsatz futuristischer Anregungen (Zukunfts-Szenarien) erreicht werden. Diese Anregungen zum Beispiel durch Zukunftsvideos [16] erhöhen die Kreativität und Vorstellungskraft der Teilnehmer. Die erfolgreiche Ideengenerierung erfordert eine kreativitätsfördernde Arbeitsatmosphäre. Mitwirkende Personen sind eher dazu bereit Ideen zu äußern, wenn Ihnen das Gefühl vermittelt wird, dass ihre Ideen aufgenommen und ernst genommen werden. Den mitwirkenden Personen müssen entsprechende Freiräume gelassen werden.

Um effektiv und effizient Anwendungsideen zu entwickeln, wird oftmals mit Fokusgruppen bzw. Gruppensitzungen gearbeitet, in denen die jeweiligen Anspruchsgruppen entsprechend repräsentiert sind [17]. Hierbei handelt es sich um einen hochgradig kollaborativen Prozess, der oftmals unterschiedliche Anspruchsgruppen einbeziehen soll [18]: Endkunden, die für den Dienst bezahlen sollen, Anwender, die mit dem Dienst umgehen sollen, die Entwickler, die ihn realisieren sollen sowie andere in den Entwicklungsprozess involvierte Rollen.

Aufgrund des geringen zeitlichen Aufwands sowie der minimalen organisatorischen und materiellen Voraussetzungen wird an dieser Stelle das Vorgehen der intuitiven Technik „Brainstorming“ erläutert. Das Brainstorming bietet trotz des geringen Aufwands eine Vielzahl an Ideen mit einer hohen Bandbreite und kann nahezu jederzeit und von jedermann (einer kleinen Gruppe) durchgeführt werden. Das Brainstorming gilt als klassische Moderationstechnik und wurde in den 1950er Jahre von Alexander F. Osborn entwickelt. Wie der Begriff (aus dem engl. für Gehirnsturm oder Ideenwirbel) vermuten lässt, steckt hinter der Technik die kreative Ideenfindung. Ziel ist die Generierung möglichst vieler Ideen. Die denkpsychologischen Blockaden sollen ausgeschalten werden, um das Unterbewusstsein anzuregen und somit die intuitiv-schöpferischen Gedanken der Gruppenteilnehmer zu fördern [19, p. 19]. Brainstorming in der Gruppe nutzt die Ideen mehrerer Personen und durch den Gedankenaustausch entstehen Synergieeffekte [20].

Um die Beteiligten von ihren festen Vorstellungen der gegenwärtigen Computernutzung zu lösen, sollten sie durch verschiedene Maßnahmen während des Workshops in ein mentales Modell der Zukunft versetzt werden, in dem die Interaktionsmöglichkeiten mit neuartigen Systemen schon vorhanden sind, so dass sie die zukünftigen Möglichkeiten begreifen und außerhalb derzeit üblicher Computer-Nutzungs-Strategien denken [21], [22]. Denkbar sind Videos und andere Stimuli, die Teilnehmer für die Dauer des Workshops aus der gegenwärtigen Realität in eine „Neue Welt“ versetzen [16]. Dieser Effekt ist auch aus Science Fiction-Filmen bekannt: Der Zuschauer hinterfragt nicht jedes „unrealistische“ Ereignis, sondern akzeptiert die Rahmenannahmen des Plots und denkt innerhalb dieses Rahmens für die Dauer des Films [16]. Dieses Phänomen sollte auf die Workshops der Ideengenerierung übertragen werden, damit die Teilnehmer im Möglichkeitsraum kreativ werden können und innovative Ideen für neue Anwendungen entwickeln.

Die Gruppenleitung bereitet Anschauungsmaterial vor und führt die Gruppe in das Problem ein, das dabei analysiert und präzisiert wird. Zudem dient es dazu, die Teilnehmer zu aktivieren. Der Input ist die Problemstellung. Es gilt darauf zu achten, dass die Frage- bzw. Aufgabenstellung weder zu breit noch zu allgemein gehalten ist [23]. Beim Brainstorming in mündlicher Form rufen die Gruppenteilnehmer nacheinander, teilweise auch durcheinander ihre Ideen dem Moderator zu. Dieser notiert

die Ideen auf dem Flipchart oder auf Kärtchen für die Metaplanwand. Der Moderator muss auf die Einhaltung der Regeln achten.

Aufgrund der Prämisse Quantität vor Qualität sollten in kürzester Zeit so viele Ideen wie möglich hervorgerufen werden. Der Gedankenfluss gewinnt an Spontaneität, die Ideen sind kurz, prägnant und ungewöhnlicher. Eine Bewertung der Ideen während des Brainstormings ist zu unterlassen. Der Phantasie soll freier Lauf gelassen werden und fremde Ideen können aufgegriffen und weiterverarbeitet werden [19, p. 20].

Wichtig ist, dass beim Brainstorming keine Ideen verloren gehen. Der Moderator sollte also darauf achten, dass sich die Gruppenteilnehmer mit ihren Ideen nicht gegenseitig überrufen. Die Ideenfindung läuft so ab, dass die Teilnehmer spontan Ideen zur Lösungsfindung nennen. Dabei regen sich die Teilnehmer gegenseitig inspirierend an, Ideen können aufeinander aufbauen und es entstehen Synergieeffekte. Die Spielregeln der Ideenfindung sind so ausgelegt, dass eine offene Atmosphäre für Kreativität geschaffen wird. So ist es ein Tabu, andere Kriterien zu kritisieren und sogenannte Totschlagargumente zu äußern. Es kommt bei der Ideenfindung auf das Sammeln von Ideen an, nicht auf deren Wertung. Sind aus Sicht der Teilnehmer bzw. des Moderators genügend Ideen in einem Zeitraum von ca. 5-30 Minuten gesammelt worden, beendet der Moderator das Brainstorming.

Zusammenfassend gelten folgende Regeln für die Ideenfindung beim Brainstorming:

- Teilnehmer nennen spontan Ideen zur Lösungsfindung.
- In einem relativ kurzen Zeitrahmen von ca. 5-30 Minuten sollen viele Ideen entstehen (Quantität vor Qualität).
- Das Kombinieren und Aufgreifen von bereits geäußerten Ideen ist ausdrücklich erlaubt und erwünscht.
- Freies Assoziieren und Phantasieren ist ausdrücklich erlaubt.
- Jede Idee wird auf z. B. einem Flipchart festgehalten.
- Die Teilnehmer inspirieren sich gegenseitig und lassen Gesichtspunkte in neue Lösungsansätze und Ideen einfließen.
- Kritik, Kommentare und Korrekturen an anderen Beiträgen, Ideen und Lösungsvorschlägen sind verboten.
- Es findet keine Wertung oder Beurteilung der Ideen statt.
- Jeder soll seine Gedanken frei äußern können nur so können die Teilnehmer ihrer Kreativität freien Lauf geben.
- Keine Totschlagargumente („So geht das nicht“, „Können Sie das verantworten?“, „Dafür sind wir nicht zuständig“, ...).
- Das Ende soll trotz der zeitlichen Begrenzung ein natürliches und kein abrupt abgebrochenes Ende sein (Ideen sind wichtiger als die Zeit).

Alternativ zu der internen Generierung ist eine externe Generierung bzw. Beschaffung von Ideen denkbar. Das kann zum Beispiel in Form von Open Innovation, Kunden- oder Experten-Workshops passieren [24].

2.1.3 Aktivität: Ideen priorisieren

Da in der ersten Phase besonders die Quantität der Ideen eine Rolle spielt, ist es in der Nachbereitung nötig, die Ideen zu priorisieren. Das Ziel der Ideen-Priorisierung ist, aus einer Vielzahl an Ideen einige wenige, aussichtsreiche Ideen zu identifizieren.

Nach einer Pause werden sämtliche Ideen (von der Gruppenleitung) vorgelesen und von den Teilnehmern bewertet und sortiert. Das bedeutet für die Teilnehmer, zunächst lediglich eine thematische Einordnung vorzunehmen sowie problemferne Ideen auszusortieren.

Im Anschluss folgt die Bewertung und Auswertung. Diese kann in derselben Diskussion durch dieselben Teilnehmer erfolgen oder von anderen Fachleuten getrennt durchgeführt werden. Dabei sollten die gegeben Rahmenbedingungen der Entwicklung beachtet werden.

Ein mögliches Tool für die Priorisierung der Ideen ist CheckMark [25]. Ein CheckMark sollte angewandt werden, wenn zum einen gewollt ist, dass die Gruppe sich auf die Punkte konzentriert, über die Einigkeit besteht und die Punkte in den Hintergrund treten sollen, bei denen sich die Gruppe uneinig ist. Zudem eignet sich CheckMark bei der Bewertung von mehr als 100 Positionen, bei denen eine quantitative Bewertung zu einer kognitiven Überlastung führen würde.

Ein CheckMark sollte nicht angewendet werden, wenn die Annahmen oder Interessen ergründet werden sollen, die den Unstimmigkeiten innerhalb der Gruppe zu Grunde liegen. In diesem Fall eignet sich StrawPoll besser. Zudem eignet sich CheckMark nicht, wenn die Abstimmung dazu genutzt werden soll, eine Diskussion zu provozieren. Denkbare Alternativen sind Broom Wagon, diese Methode zeichnet sich damit aus, auf schnelle Weise Schlüsselergebnisse herauszustellen und PopcornSort, womit schnell und effektiv Brainstorming Ergebnisse in Cluster zusammengefasst werden [25].

Im Rahmen von CheckMark erhält jeder Teilnehmer die Möglichkeit, aus einer Liste mit Positionen seine Favoriten auszuwählen, indem er diese mit einem Häkchen kennzeichnet. Normalerweise wird dabei die Anzahl der Häkchen begrenzt. Die in Teilaktivität I erstellte Ideenliste dient als Input für die CheckMark Methode. Der Output ist in diesem Fall eine Liste der Ideen, die nach Präferenzen der Teilnehmer geordnet ist.

Die Durchführung gliedert sich in zwei Arbeitsschritte, durch die der Moderator die Teilnehmer begleitet. Jeder der Teilnehmer findet vor sich eine Liste mit allen Ideen, die im Rahmen des Brainstormings erarbeitet wurden. Die Teilnehmer lesen Sie sich die Liste durch und markieren alle Ideen, die sie favorisieren mit einem Häkchen. Dabei ist von den Teilnehmern zu beachten, dass die Häkchen auf die zuvor festgelegte Anzahl begrenzt wurde, so dass nur eine bestimmte Anzahl an Ideen von jedem Teilnehmer priorisiert werden können.

Haben alle Teilnehmer die Auswahl beendet, berichtet der Gruppenleiter über die Resultate. Dieses kann beispielsweise folgendermaßen aussehen: „So wie es aussieht sind X von Ihnen der Meinung, dass das Item A das wichtigste auf der Liste ist. Y von Ihnen sind der Meinung, dass das Item B eine weitere Betrachtung verdient. […]“.

CheckMark besticht aus zwei Gründen. Zum einen ist es möglich, die Übereinstimmungen innerhalb der Gruppe offen zu legen, ohne dass die Unstimmigkeiten dediziert aufgezeigt werden. Herrschen innerhalb der Gruppe Konflikte, so wird den Teilnehmern bewusst, wo Ihre Gemeinsamkeiten liegen,

ohne dass sie dabei zu viel Aufmerksamkeit auf Punkte legen müssen, die nicht konsensfähig sind. Zum anderen bedingt CheckMark eine relativ geringe kognitive Last. Es ist wesentlich einfacher ein Häkchen zu setzen, als die einzelnen Positionen über eine quantitative Skala zu bewerten. Insbesondere bei einer Liste mit vielen Positionen (die beim Brainstorming optimaler Weise resultiert) kommt hier das Besondere von CheckMark zum Tragen.

2.1.4 Werkzeugunterstützung: GSS

Oft sind Prozesse mit einer Vielzahl von Beteiligten ineffizient, weil die Kommunikationsprozesse in großen Gruppen langsam verlaufen und nur unbefriedigende Ergebnisse entstehen. Daher bietet sich gerade in diesen für den Innovationsprozess wichtigen frühen Phasen der Einsatz von Gruppenunterstützungssystemen (auch Electronic Meeting Systems (EMS) oder Group Support Systems (GSS)) zur Verbesserung des Prozesses an [17]. GSS wurden entwickelt, um die Effizienz und Effektivität von Sitzungen zu verbessern, indem den Teilnehmern eine Vielzahl von Werkzeugen für die Gruppenunterstützung angeboten wird. Auch der Erfolg derartiger Systeme im Sinne von Kostenreduktion oder schnellerer Durchführung ist in frühen Untersuchungen oftmals belegt worden [26] [27]. Wird eine Sitzung mit einem GSS unterstützt, so verändert dies die Möglichkeiten der Teilnehmer, miteinander zu kommunizieren, den Sitzungsprozess zu strukturieren und Informationen in der Sitzung zu verarbeiten [28]. Die wesentlichen Funktionen von GSS lassen sich in diesen drei Kategorien zusammenfassen [17]:

- Kommunikationsunterstützung: Der zentrale Ansatzpunkt von GSS zur Verbesserung der Gruppenkommunikation ist das Ermöglichen von paralleler und anonymer Kommunikation von Sitzungsbeiträgen. Durch Parallelisierung der Kommunikation können alle Teilnehmer unabhängig voneinander ihre Beiträge übermitteln oder andere Beiträge bearbeiten, wobei sie alle Änderungen und Ergänzungen der anderen Teilnehmer sehen. Nicht in allen Sitzungsphasen ist es erforderlich, alle Teilnehmer in einer streng sequentiellen Form zu Wort kommen zu lassen. Die Parallelisierung der Arbeit kann so Zeit sparen. Durch die Anonymisierung ist der Urheber eines Beitrages für die anderen Sitzungsteilnehmer nicht zu erkennen. Dies wird durch den Einsatz von Computern erheblich erleichtert, da so kein Sprecher und keine Handschrift identifizierbar sind. Davon verspricht man sich eine sachlichere und offenere Kommunikation in Gruppen mit großen Status- oder Hierarchieunterschieden.
- Prozessstrukturierung: GSS unterstützen in besonderem Maße eine ergebnisorientierte Strukturierung des Sitzungsprozesses. Zum einen stellen sie oftmals Werkzeuge zur Festlegung und Durchsetzung einer definierten Folge von Sitzungsaktivitäten zur Verfügung. Zum anderen gibt es Werkzeuge für einzelne Sitzungsphasen, die ein in bestimmter Weise strukturiertes Vorgehen der Sitzungsteilnehmer erfordern oder nahe legen.
- Informationsverarbeitung: Die Verbesserung der in Sitzungen erzeugten Informationen ist eine der zentralen Leistungen von GSS. Da Beiträge der Teilnehmer elektronisch erfasst werden, können die in einer Sitzungsphase gewonnenen Informationen in anderen Phasen und über die Sitzung hinaus weiterverwendet werden. Die Folge der Aktivitäten, die Beiträge der Gruppenmitglieder, Abstimmungsergebnisse und Zusammenfassungen können elektronisch oder auf Papier den Teilnehmern zur Verfügung gestellt werden. Weiterhin lassen sich durch die Computerunterstützung bestimmte Analyseschritte beschleunigen. So können schnell Fragebögen ausgewertet oder Abstimmungsergebnisse ermitteln werden.

2.2 VB: Nutzungskontext verstehen

Besonderheiten des Nutzungskontexts in der VENUS-Entwicklungsmethode:

- Umfasst auch rechtliche Rahmenbedingungen und rechtliche Beschreibungslücken
- Umfasst auch gesellschaftliche Vorbehalte
- wird zum notwendigen Element für sozialverträgliche Softwareentwicklung

Produkt	Nutzungskontext
Verantwortliche Rolle	Anforderungsanalyst
Mitwirkende Rollen	Diverse

Die DIN EN ISO 9241-210 beschreibt die Ermittlung des Nutzungskontextes wie folgt: „Die Benutzermerkmale, Arbeitsaufgaben und die organisatorische, technische und physische Umgebung bestimmen den Kontext, in dem das System verwendet wird. Es ist sinnvoll, Informationen zum aktuellen Kontext zu sammeln und zu analysieren, um denjenigen Kontext zu verstehen und anschließend festzulegen, der für das zukünftige System gelten wird. Durch die Analyse bestehender oder ähnlicher Systeme (gegebenenfalls einschließlich manueller Systeme) können, sofern nach wie vor zutreffend, Informationen zum gesamten Spektrum der Kontextprobleme gewonnen werden, einschließlich Mängel und Ausgangsniveaus für Leistung und Zufriedenstellung. Dadurch können Notwendigkeiten, Probleme und Einschränkungen aufgezeigt werden, die ansonsten möglicherweise übersehen würden, vom zukünftigen System jedoch abgedeckt werden müssen“ [29, p. 20].

Das Verständnis des Nutzungskontextes wird in der VENUS-Entwicklungsmethode nicht ausschließlich mit dem Ziel aufgebaut, die Gebrauchstauglichkeit sicherzustellen. Zusätzlich dient das Wissen über die Begleitbedingungen dazu, eine rechtsverträgliche und vertrauenswürdige Gestaltung zu erreichen. Zu diesem Zweck wurde die normative Definition des Nutzungskontextes erweitert. Ohne den Nutzungskontext verstanden zu haben, ist es nicht möglich, sozialverträgliche Software zu entwickeln. Die Analyse des Nutzungskontextes sollte daher ein zentraler Bestandteil zu Beginn der sozialverträglichen Softwareentwicklung sein.

Welche Methoden sich für ein bestimmtes Projekt eignen hängt davon ab, wie die grundsätzlichen Bedingungen sind, d.h. stehen Nutzer zur Verfügung und/oder gibt es Experten, die sich bereits im Vorfeld mit dem Produkt beschäftigen können? Wie sind die Eigenheiten der Nutzer, des Produktes oder die Fähigkeiten der einzelnen mitwirkenden Personen einzuschätzen? Gibt es zeitliche oder finanzielle Einschränkungen? ISO/TR 16982 schlägt ferner einige Methoden vor, die geeignet sind, um den Nutzungskontext (auch unabhängig von Gebrauchstauglichkeit) zu bestimmen [30, p. 9].

2.2.1 Produkt: Nutzungskontext

Die DIN EN ISO 9241-210 fordert für die Beschreibung des Nutzungskontextes folgende obligatorische Inhalte:

- **Nutzer und sonstige Interessengruppen:** Es kann ein Spektrum unterschiedlicher Nutzergruppen sowie weiterer Interessengruppen geben, deren Erfordernisse wichtig sind. Relevante Gruppen müssen identifiziert und ihre Beziehung zur vorgeschlagenen Entwicklung in Form von wesentlichen Zielen und Einschränkungen beschrieben werden.

- **Merkmale der Nutzer oder Nutzergruppen:** Wesentliche Merkmale der Nutzer müssen identifiziert werden. Diese können Kenntnisse, Fertigkeiten, Erfahrung, Ausbildung, Übung, physische Merkmale, Gewohnheiten, Vorlieben und Fähigkeiten einschließen. Falls es erforderlich ist, sollten die Merkmale verschiedener Nutzertypen definiert werden, z. B. Nutzer mit unterschiedlicher Erfahrung oder körperlichen Fähigkeiten.

- **Ziele und Arbeitsaufgaben der Nutzer:** Die Ziele der Nutzer und die Gesamtziele des Systems müssen identifiziert werden. Die Merkmale der Arbeitsaufgaben, die die Gebrauchstauglichkeit und Zugänglichkeit beeinflussen können, müssen beschrieben werden, z. B. die Art, in der Nutzer typischerweise Arbeitsaufgaben ausführen, die Häufigkeit und die Zeitdauer für die Ausführung, wechselseitige Abhängigkeiten und parallel auszuführende Tätigkeiten. Falls es mögliche nachteilige Auswirkungen auf die Gesundheit und Sicherheit gibt (z. B. übermäßige Arbeitsbelastung durch ein unangemessenes Tempo in einem Call-Center) oder wenn die Gefahr besteht, dass die Arbeitsaufgabe fehlerhaft abgeschlossen werden könnte (z. B. Tätigen eines falschen Einkaufs), sollten diese ebenfalls ermittelt werden. Arbeitsaufgaben sollten nicht nur in Bezug auf Funktionen und Merkmale beschrieben sein, die durch ein Produkt oder System zur Verfügung gestellt werden.

- **Umgebung(en) des Systems:** Die technische Umgebung einschließlich Hardware, Software und Materialien muss identifiziert werden. Außerdem müssen die relevanten Merkmale der physikalischen, sozialen und kulturellen Umgebungen beschrieben werden. Zu den physikalischen Eigenschaften zählen Aspekte wie beispielsweise thermische Bedingungen, Beleuchtung, Raumgestaltung und Möbel. Zu den sozialen und kulturellen Aspekten der Umgebung zählen Faktoren wie Arbeitsweisen, Organisationsstruktur und Einstellungen.

Für die Entwicklung von ubiquitären Systemen, deren Ziel eine hohe Sozialverträglichkeit ist, sind zusätzlich folgende Punkte wichtig:

- **Rechtliche Rahmenbedingungen:** Die Gesetze und Rechtsnormen, die auf das System anzuwenden sind, müssen geklärt werden. Es muss geklärt sein, in welchen Ländern und in welchem Zusammenhang, z. B. privat oder beruflich, die Software eingesetzt werden soll. Zudem ist zu klären, ob Dokumentationspflichten für die Beweisbarkeit möglicher Vergehen bestehen und welche Ansprüche vom Gesetzgeber an die Datensicherheit gestellt werden. Zusätzlich sind bestehende rechtsfreie Räume zu identifizieren, die in der normativen Anforderungserhebung Recht (vgl. Abschnitt 3.1) berücksichtigt werden müssen.

- **Gesellschaftliche Vorbehalte:** In der Nutzergruppe sollten bestehende Vorbehalte und Bedenken gegenüber der geplanten Anwendung identifiziert werden und ob sich diese auf Technologien, auf die Entwickler oder auf die Art der Anwendung im Allgemeinen beziehen. Zudem sollten Ängste erhoben werden, die zu einem Ausschluss bestimmter Nutzergruppen von der Serviceleistung führen können.

Der Nutzungskontext des Systems sollte ausreichend detailliert beschrieben sein, um die Aktivitäten in Bezug auf die Anforderungen, die Gestaltung und die Bewertung zu unterstützen. Die Beschreibung des Nutzungskontexts stellt ein Dokument dar, das während des Entwicklungsprozesses überarbeitet, erweitert und aktualisiert wird. In einer Anfangsphase der Entwicklung könnte es zum Beispiel lediglich möglich sein, Aufgabenziele zu bestimmen, während detaillierte und aufgabenspezifische Tätigkeiten erst später in den Fokus genommen werden.

Die Erhebung des Nutzungskontextes kann entweder in einem Workshop oder mit Hilfe von Interviews erhoben werden. Ein Workshop bietet Ergebnisse innerhalb kurzer Zeit. Die Interviews ermöglichen ein tieferes Verständnis, da in diesen detaillierter auf den Nutzungskontext der einzelnen Personen eingegangen werden kann, beanspruchen jedoch mehr Zeit.

2.2.2 Aktivität: Workshop

Die Möglichkeit zur Analyse des Nutzungskontextes besteht innerhalb eines Workshops mit verschiedenen Stakeholdern. Dieses Treffen sollte durch den Anforderungsanalyst strukturiert und gut vorbereitet werden. Bei diesem Meeting soll die Beschreibung des Nutzungskontextes vorbereitet werden. Der Anforderungsanalyst sollte also versuchen, genau diese Inhalte in der Diskussions-Runde anzusprechen und zu erfassen. Folgende Leitfragen können helfen [31]:

- Wer ist die Zielgruppe?
- Was sind die Aufgaben der Zielgruppe?
- Welche Werkzeuge stehen der Zielgruppe zur Verfügung, um die Aufgaben zu erledigen?
- Was ist die physische Umgebung des zu entwickelnden Systems?
- Welche sozialen und organisatorischen Strukturen gilt es zu berücksichtigen?
- Was sind die technischen Rahmenbedingungen? Welche technischen Mittel gibt es?
- Welche anderen Faktoren beeinflussen die zukünftigen Nutzer?
- Welche rechtlichen Rahmenbedingungen bestehen?
- Gibt es Ängste und Vorbehalte gegenüber der Anwendung?

2.2.3 Aktivität: Interview

Um den Nutzungskontext zu verstehen, können die identifizierten Nutzer in Interviews befragt werden. Die Interviews werden durch einen Leitfragenkatalog strukturiert. Im Anschluss erfolgt die Umsetzung der erfragten Inhalte in Kontextszenarien, die den Nutzungskontext in einem Fließtext beschreiben. Dieser Fließtext wird in Sinneinheiten gegliedert und es wird für jede Sinneinheit nach einem zugrundeliegenden Erfordernis gesucht.

Die Leitfragen für die durchzuführenden Kontextinterviews sind auf die jeweiligen Rahmenbedingungen der ubiquitären Anwendung zu anzupassen. Folgende Liste mit Leitfragen basiert auf einer Überarbeitung bestehender Leitfragenlisten aus der Literatur [32] [33]. Die Überarbeitung [34] erweitert die Fragebögen um Erkenntnisse aus der Activity Theory [35], um die unterschiedlichen Aspekte des Nutzungskontextes in einem sinnvollen Gesamtzusammenhang zu verstehen. Die Activity Theory liefert einen strukturierten Erklärungsrahmen für das Verhalten von Menschen und ermöglicht es, den Nutzungskontext umfassend und losgelöst von den Paradigmen der klassischen Mensch-Computer-Interaktion zu berücksichtigen. Somit erfolgt ein Abgleich, ob alle für UC relevanten Aspekte (z. Bsp. andere Werkzeuge) berücksichtigt wurden. Die Leitfragen sind ggf. mit dem Auftraggeber abzustimmen.

- Bitte erläutern Sie (beispielhaft) auf welche Weise die zu unterstützende Aufgabe durchgeführt wird. Erzählen Sie dabei, wie und ob der Ablauf variieren kann.
- Aus welchen Teilen ist die zu unterstützende Aufgabe zusammengesetzt?
- Welche Qualifikation und welche Vorkenntnisse sind für die Durchführung der zu unterstützenden Aufgabe erforderlich?
- Welche Rollen nehmen die Personen ein? Was ist Ihre Rolle?

- Welche expliziten und impliziten Regeln, Normen und Vorgänge beeinflussen die Art und Weise, wie die zu unterstützende Aufgabe durchgeführt wird?
- Inwiefern ist die Interaktion mit anderen Personen geregelt?
- Wann sind Sie mit der Durchführung zufrieden bzw. unzufrieden?
- Welche Störungen und Konflikte gibt es zwischen dem Aufgabenziel und anderen Zielen?
- Wie gehen Sie damit um, wenn bei der Durchführung der zu unterstützenden Aufgabe Probleme auftreten?
- Wie könnten die Ziele der zu unterstützenden Aufgabe auf bessere Weise erreicht werden?
- Welche wichtigen Sonderfälle müssen berücksichtigt werden?
- Welche Werkzeuge verwenden Sie bei der Durchführung der zu unterstützenden Aufgabe?
- Wann verwenden Sie welche Werkzeuge und wie organisieren Sie diese?
- Wie werden die Werkzeuge miteinander geteilt?
- Wie schwer war es für Sie, den Umgang mit den Werkzeugen zu lernen? Was hätte einfacher sein können?
- Wie passen die Werkzeuge zu dem Ablauf bei der Durchführung der zu unterstützenden Aufgabe?
- Wie beeinflussen Ihre Werkzeuge die Durchführung der zu unterstützenden Aufgabe?
- Wie beeinflussen die genutzten Werkzeuge die Art und Weise, wie Sie über die Durchführung der zu unterstützenden Aufgabe denken und urteilen?
- Welche Anteile der zu unterstützenden Aufgabe werden von Ihnen mental durchgeführt?
- Wie haben frühere, von Ihnen verwendete Systeme die Art und Weise der Zusammenarbeit beeinflusst? Welchen Einfluss auf die Tätigkeit hatte der Wechsel von älteren zu neueren Werkzeugen?

Beispiel Nutzungskontext Meet-U

Für die Beispielapplikation Meet-U wurde der Nutzungskontext innerhalb eines Workshops erhoben. Das Ergebnis sieht wie folgt aus:

- Wer ist die Zielgruppe? Junge Menschen bis 30 Jahre, die sich mit ihren Freunden und Bekannten vernetzen wollen.
- Was sind die Aufgaben der Zielgruppe? Events planen, Freunde finden, den Weg zum Event finden, Kommunikation mit Freunden und Eventteilnehmern
- Welche Werkzeuge stehen der Zielgruppe zur Verfügung, um die Aufgaben zu erledigen? Smartphone
- Was ist die physische Umgebung des zu entwickelnden Systems? Überall. Innen und außen. Zu jeder Tageszeit.
- Welche sozialen und organisatorischen Strukturen gilt es zu berücksichtigen? Freundesgruppen. Einladen von Fremden. Blinddating
- Was sind die technischen Rahmenbedingungen? Welche technischen Mittel gibt es? Lokalisierung mit Smartphone. Internetverbindung. Zentraler Server. Kleiner Bildschirm.
- Welche anderen Faktoren beeinflussen die zukünftigen Nutzer? Soziale Vernetzung. Gruppendynamik.
- Welche rechtlichen Rahmenbedingungen bestehen? Schutz der Privatsphäre des Einzelnen. Recht auf informationelle Selbstbestimmung. Vertragsrecht z. B. bei kommerziellen Events.

- Gibt es Ängste und Vorbehalte gegenüber der Anwendung? Speicherung von Profildaten. Sicherheit von Information, insbesondere bei Blinddating. Verschlüsselte Übertragung der Daten.

2.3 VB: Anwendungsziele erarbeiten

Besonderheiten der Erarbeitung der Anwendungsziele in der VENUS-Entwicklungsmethode:

- (keine)

Produkt	Anwendungsziel
Verantwortliche Rolle	Anforderungsanalyst
Mitwirkende Rollen	Diverse

Ziele konkretisieren die Anwendungsidee des zu entwickelnden Systems [36]. Sie können natürlichsprachlich dokumentiert werden. Hierzu können Ziele und die zugehörigen Kontextinformationen in einer Zielschablone festgehalten werden [37]. Um ein prägnant formuliertes Anwendungsziel zu erarbeiten, bedarf es der intentionalen Beschreibung der charakteristischen Merkmale des zu entwickelnden Systems. Dabei werden die Anwendungsziele aus Nutzersicht erarbeitet. Die Verfeinerung der Vision wird auch als Zieldekomposition bezeichnet. Es besteht in der Regel zunächst eine Vision und damit eine grobe Vorstellung der Anwendung. Je detaillierter und konkreter die Vorstellung wird, desto konkreter können Ziele vereinbart und vereinheitlicht werden. Diese Verfeinerung oder auch Zieldekomposition ist eine Zerlegung in Teilziele [38]. Zu Beginn sollte zudem eine vollständige Erfassung aller Attribute für ein Ziel vermieden werden. Anfangs sind die Basisattribute von höherer Bedeutung, also Bezeichner, Name, Quelle, Verantwortlicher und die Zielbeschreibung. Anschließend werden für jedes Ziel die Beziehungen zu anderen Zielen definiert. Danach sollten die Ziele auf Vollständigkeit und korrekte Beziehungen überprüft werden. Ist das nicht der Fall, sollten fehlende Ziele und Zielbeziehungen ergänzt werden. Zum Abschluss der Zieldokumentation müssen diese durch fehlende Attribute vervollständigt werden [37]. Zur Formulierung der Anwendungsziele können nach [37] die folgenden sieben Regeln unterstützen:

- Prägnante Formulierung. Vermeiden von Füllwörtern oder Allgemeinplätzen.
- Verwendung von Aktivformulierungen. Vermeidung von Passivformulierungen.
- Formulierung von möglichst überprüfbaren Zielen
- Verfeinerung von nicht überprüfbaren Zielen. Durch die Dekomposition eines nicht überprüfbaren Ziels in überprüfbare Ziele wird die Überprüfbarkeit wiederhergestellt.
- Formulierung des Mehrwerts eines Ziels auf möglichst präzise Art und Weise.
- Begründung für das angegebene Ziel.
- Vermeidung von Lösungsansätzen, es sei denn, der angegebene Lösungsansatz ist verbindlich.

Beispiel Anwendungsziel Meet-U

Was ist es?

Eine Applikation für mobile Endgeräte, welche im Hintergrund Echtzeitinformationen verwertet und somit dem Nutzer Services im sozialen Miteinander mit anderen Nutzern bietet.

Was tut es?

Neben dem Organisieren von Verabredungen mit Freunden, welche gleiche Interessen pflegen, navigiert die Applikation zu dem gewünschten Treffpunkt, versorgt den Nutzer mit Echtzeitinformationen, abhängig von Ort, Zeit und anderen Nutzern der Applikation Meet-U und dient als persönlicher Assistent für das Organisieren von Freizeitaktivitäten und Events. Das bedeutet exemplarisch, dass Meet-U dem Nutzer Freizeitaktivitäten und Events vorschlägt, die den persönlichen Interessen des Nutzers entsprechen.

Was ist der Nutzen?

Es erleichtert dem Nutzer das Organisieren von Freizeitaktivitäten mit anderen Meet-U Nutzern und kann durch den mobilen Charakter der Applikation zum Teil komplexe Echtzeitinformationen verarbeiten und dem Nutzer übersichtlich und brauchbar zur Verfügung stellen. Zudem erleichtert die Applikation durch die Suchfunktion beispielsweise das Auffinden eines Restaurants, also bedient es den Nutzen des Ausgleiches eines Informationsdefizites des Nutzers.

2.4 VB: Personas erarbeiten

Besonderheiten der Erarbeitung der Personas in der VENUS-Entwicklungsmethode:

- Spezifische (Standard-)Personas bezogen auf Akzeptanz, Vertrauen und Rechtsverträglichkeit

Produkt	Persona-Beschreibungen
Verantwortliche Rolle	Usability Engineer
Mitwirkende Rollen	-

Personas sind ursprünglich eine Methode des Usability-Engineering und wurden von Alan Cooper [39] für die nutzerzentrierte Gestaltung von Technologien entwickelt. Es werden verschiedene Arten an Personas unterschieden. Die „primäre Persona“ bestimmt den Fokus bei der Entwicklung, ihre Bedürfnisse müssen unbedingt berücksichtigt werden. Für ein Entwicklungsprojekt sollte pro Nutzergruppe eine Primär-Persona formuliert werden. Für UC-Systeme ist die Berücksichtigung der Nutzerakzeptanz von besonderer Bedeutung. Aus diesem Grund sollten zwei zusätzliche Persona-Beschreibungen formuliert werden: Eine Persona sollte dabei eine sehr hohe Wahrscheinlichkeit der Akzeptanz des Systems aufweisen, und die andere Persona eine besonders geringe Wahrscheinlichkeit der Akzeptanz des neuen Systems (vgl. Abschnitt 2.4.2). Insgesamt sollte die Anzahl der Personas überschaubar bleiben: mehr als 5 unterschiedliche Personas sind nicht zu empfehlen. Zusätzlich kann eine sogenannte Non-Persona nützlich sein um zu beschreiben, wer die zukünftigen Nutzer nicht sind.

2.4.1 Produkt: Persona

Eine Persona-Beschreibung ist eine steckbriefartige, prägnante Darstellung eines archetypischen Nutzers, die mit ihrer spezifischen Charakterisierung stellvertretend für die gesamte Nutzergruppe steht. Die Personas dienen dem Entwicklerteam dazu, die Anwendungsszenarien mit konkreten, zukünftigen Nutzern plastisch, realitätsnah und nachvollziehbar für alle Beteiligten zu machen und die folgenden Entwicklungsschritte zu unterstützen. Das Besondere an einer Persona-Beschreibung ist, dass sie, im Gegensatz zu anderen formalen Nutzergruppen-Beschreibungen, den Nutzer explizit als Individuum darstellt.

Jede Persona-Beschreibung sollte einzigartig sein und sich nicht zu stark mit einer anderen überschneiden. Die Form einer Persona-Beschreibung ist der Steckbrief. Die zu konkretisierenden Bestandteile einer Persona-Beschreibung sind:

- Name
- Foto
- Alter
- Beruf
- Ziele
- Wissen und Bildung
- Fähigkeiten bzw. Einschränkungen wie z. B. körperliche und geistige

Personas sind fiktiv und dürfen keine bekannten Personen darstellen. Mit Inhalten in Form von Zitaten kann eine Persona „für sich selbst sprechen“. Dies ist insbesondere für den Bestandteil Ziele

passend. Stereotypische Charaktere können wertvolle Hinweise für eine Persona-Beschreibung liefern.

2.4.2 Werkzeugunterstützung: Standard-Personas

Forschungen in VENUS weisen auf den Zusammenhang zwischen Persönlichkeitsmerkmalen und der Akzeptanz, dem Vertrauen und der wahrgenommenen Rechtsverträglichkeit hin. Diese Erkenntnisse dienen dazu die Eigenschaften von zwei Personas zu definieren: Die erste Persona repräsentiert eine potentielle Nutzergruppe, bei der eine positive Bewertung des Systems sehr wahrscheinlich ist. Die zweite Persona-Beschreibung passt zu einer potentiellen Nutzergruppe, bei der eine ablehnende Haltung zu erwarten ist. Die entsprechenden Personas werden beschrieben, um bei der Entwicklungsarbeit im Sinne einer sozialverträglichen Gestaltung die Bedürfnisse der kritisch eingestellten Personen gezielt zu berücksichtigen, ohne dabei die Bedürfnisse der zugewandten Personengruppe außer Acht zu lassen. [40] Für die Umsetzung der aufgeführten Informationen in Texte für Persona-Beschreibungen ist es hilfreich, Vokabellisten zu nutzen, die für die jeweiligen Persönlichkeitsmerkmalen passen. Folgende zwei Personas (weitere in der zugehörigen Quelle) nutzen die Erkenntnisse und definieren ablehnende und akzeptierende Persönlichkeiten [40]:

Potentielle Ablehnerin (Emotionale Stabilität niedrig und wahrgenommene Einfachheit der Nutzung niedrig): Martha ist 31 Jahre alt und studiert noch. Sie befürchtet, dass sie ihre abschließenden Prüfungen an der Universität nicht schaffen wird. Martha fühlt sich schuldig, in ihrem Alter noch nicht in der Lage zu sein, auf eigenen Füßen zu stehen. Martha ist schüchtern. Als sie von Meet-U gehört hat, hatte sie die Hoffnung, dass die Anwendung ihr helfen könnte, Verabredungen zu treffen und neue Freunde zu finden. Als sie Meet-U ausprobiert hat, war ihr die Anwendung allerdings viel zu kompliziert und sie war verwirrt von den angebotenen Funktionen.

Potentieller Akzeptierer (Gewissenhaftigkeit niedrig und wahrgenommene Nützlichkeit hoch): Michael ist 24 Jahre alt und lebt noch bei seinen Eltern. Er studiert Betriebswirtschaft und lässt sich beim Studieren Zeit. Eigentlich müsste er für eine Prüfung lernen, aber er hat die Unterlagen verlegt und entscheidet sich, die Prüfung dann im nächsten Semester zu machen. An Meet-U mag Michael, dass ihm die Anwendung hilft, sich zu organisieren. Oft vergisst er Verabredungen oder sagt Freunden nicht Bescheid, wenn er später kommt.

Beispiel Persona für Meet-U: Alice

Alice ist 24 Jahre alt und Studentin. Sie interessiert sich für neue Technologien, wobei sie jedoch nicht jedem neuen Trend folgt. Seit einem Jahr besitzt sie ein modernes Smartphone mit Touchscreen. Sie nutzt es zum Telefonieren und um SMS zu schreiben und verwendet auch verschiedene Apps, die ihren Alltag unterstützen, wie z. B. einen Terminkalender, in dem sie Treffen mit Freunden einträgt. Ihrer Meinung nach unterstützt sie der Terminkalender aber nur rudimentär zur Organisation ihrer privaten Treffen. Außerdem lädt sie, wenn sie von einer coolen App gehört hat, diese herunter und probiert sie aus. Das passierte auch bei Meet-U, dessen kostenlose Version sie nun ausprobiert. Wenn ihr diese praktisch erscheint, würde sie auch die zu bezahlende, werbefreie Version herunterladen. Alice ist auch Mitglied sozialer Netzwerke wie z. B. Facebook, auf die sie auch über ihr Smartphone zugreift. Daneben nutzt sie das Smartphone auch für die Informations- und Nachrichtensuche. Zitat von Alice zur App: „Ich möchte eine App, die mich nicht unnötig ablenkt und macht, was ich möchte."

2.5 VB: Anwendungsszenarien erarbeiten

Besonderheiten der Erarbeitung der Anwendungsszenarien in der VENUS-Entwicklungsmethode:

- Parallele Entwicklung eines (rechtsverträglichen) Geschäftsmodells
- Überprüfung der Nutzerakzeptanz mittels des Anwendungsszenarios

Produkt	Anwendungsszenarien
Verantwortliche Rolle	Anforderungsanalyst
Mitwirkende Rollen	Diverse

Die Erarbeitung von Anwendungsszenarios hat zum Ziel, die Anwendung im Nutzungskontext ganzheitlich auf abstraktem Niveau zu beschreiben. Anwendungsszenarien konkretisieren die Idee für die Anwendung (vgl. Abschnitt 2.1). Die gemeinsame Arbeit an den Szenarien dient dem Austausch zwischen den beteiligten Personen und Disziplinen. Es entsteht ein gemeinsames Problembewusstsein und der Raum möglicher Lösungen wird interdisziplinär aufgespannt und diskutiert. Die stattfindende Kommunikation fördert das gegenseitige Verständnis. Der gefundene Konsens für das große Ganze der Anwendung wird im Anwendungsszenario fixiert und stellt im weiteren Verlauf ein Bezugssystem dar. Auf diese Weise können sich wiederholende Diskussionen vermieden werden und die Gefahr, sich in Details zu verlieren wird gemindert. Die Anwendungsszenarien helfen bei der Identifikation von Anforderungen. Aus einem Anwendungsszenario lassen sich für die Phase Konzeptdesign Anwendungsfälle generieren, welche die jeweils erforderlichen Interaktionen zwischen Mensch und System konkret erklären. Die Anwendungsszenarien werden aus Sicht einer Persona als Repräsentant einer Nutzergruppe formuliert, welche ein konkretes Ziel verfolgt. Eine Validierung der Szenarien mit Nutzern und die Überprüfung der technischen Realisierbarkeit werden durchgeführt, um die zuvor entwickelten Ideen zu überprüfen. Ziel sind technisch machbare Szenarien, deren Kern die Ziele der späteren Nutzer darstellen. Das zu erstellende Geschäftsmodell dient der Überprüfung der Marktfähigkeit der Anwendung. Als Erweiterung der Szenarien sollten im Geschäftsmodell Informationen enthalten sein, die für eine Gestaltung der Technik notwendig sind.

2.5.1 Produkt: Anwendungsszenario

Anwendungsszenarien sind auf Basis der Vorgehensbausteine Ideen, Anwendungsziele und den zuvor festgelegten Personas zu erarbeiten. Insbesondere ist die Orientierung an den Zielen, Bedürfnissen und Eigenschaften der Personas erforderlich [41]. Ein Anwendungsszenario beschreibt hierzu beispielhaft die Anwendung des Systems in einem konkreten Fall aus Sicht einer Persona. Die Persona verfolgt dabei ein übergeordnetes Ziel, welches sie durch Einsatz und Unterstützung des Systems zu erreichen versucht. In die Szenario-Beschreibung fließen die Ergebnisse aus den Befragungen von Nutzern zum Verständnis des Nutzungskontextes mit ein (vgl. Baustein Nutzungskontext verstehen). Das Szenario muss ein realistisches Beispiel der Interaktion mit dem geplanten System bis zur Zielerreichung abbilden.

Ein Szenario hat in Anlehnung an die von Richter und Flückinger [41] genannten Szenarien-Merkmale sowie aufgrund der für die nutzerorientierte Gestaltung [42] zu beachtenden Komponenten des Nutzungskontextes, folgende Qualitätsmerkmale:

- **Persona-bezogen**: Ein Szenario „…ist für eine bestimmte [Persona] entworfen, berücksichtigt ihre Eigenschaften und erfüllt ihre Bedürfnisse.“ [41].
  Beispiel: Die Persona Alice ist in das Szenario integriert und das Szenario ist auf eines ihrer Ziele und ihre Bedürfnisse maßgeschneidert.
- **Anwendungsfallbezogen**: Ein Szenario „…stellt einen konkreten Fall aus der Anwendung dar.“ [41]. Dabei werden alle dafür notwendigen Aufgaben berücksichtigt und beschrieben [42] . Das Szenario ist dabei abgeschlossen und der Zustand vor und nach dem Szenario wird formuliert.
  Beispiel: Alice möchte mit Freunden am Samstagabend ins Kino gehen und möchte für die Planung die neue App Meet-U verwenden. Sie erstellt ein privates Event, sie lädt Freunde ein, usw. Der Kinobesuch findet statt.
- **Realitätsnah**: Ein Szenario „…zeigt wie die [Persona] das System im realen Umfeld einsetzen [wird].“ [41]. Im Szenario werden auch Informationen zu parallelen Aktivitäten neben der Aufgabenbewältigung mit dem System gegeben oder weitere Hilfsmittel neben dem System genannt, die zur Erreichung des Ziels von den Personas eingesetzt werden [42]. Zudem wird die soziale und physikalische Umgebung definiert [42].
  Beispiel: Meet-U wird im Kinosaal stumm geschaltet, da es dort nicht gern gesehen wird, wenn Mobilgeräte störende Geräusche machen.
- **Lösungsbezogen:** Ein Szenario „…illustriert die für die Entwicklung der neuen Lösung relevanten Aspekte.“ [41].
  Beispiel: Das Kino bietet einen Ticket-Service an, den es auf diese Weise aktuell noch nicht gibt. Smartphones werden als Plattform für die Benutzungsschnittstelle angeführt.
- **Besonderheiten einbeziehend:** Ein Szenario „…beschränkt sich nicht auf einen Schönwetterfall, sondern beschreibt auch exemplarisch wichtige Ausnahme- und Fehlersituationen.“ [41].
  Beispiel: Peters Auto hat einen platten Reifen, weshalb er kurzfristig absagen muss.

Szenarien sind als ein textlicher Entwurf des Systems und der allgemeinen Interaktionen und Abläufe des Nutzers mit dem System zu verstehen, ohne dabei jedoch eine konkrete Gestaltung der Benutzungsschnittstelle oder der technischen Funktionsweise zu definieren. Hier können zudem bereits alle der Bedarfsanalyse entstammenden Rahmenbedingungen aufgeführt werden. So können etwa zum Einsatz kommende Ein- und Ausgabegeräte oder die in Frage kommenden Funkschnittstellen einbezogen werden.

Die Formulierung der Szenarien erfolgt in einfachen Sätzen und auf einem Abstraktionslevel, das alle beteiligten Entwicklerdisziplinen verstehen. Die technische Gestaltung soll nicht vorweg genommen werden. Es geht darum zu definieren welche „Probleme“ die Technik später lösen soll und nicht darum wie sie dies tut. Bezogen auf die einzelnen Disziplinen stellt dies häufig eine Herausforderung dar. So muss für einen Usability Engineer in den Szenarien die Interaktion der Persona mit dem System festgehalten sein, ohne dabei allerdings technische oder gestalterische Entscheidungen vorwegzunehmen. Ähnliches gilt für den Juristen: Die Szenariobeschreibung ist für ihn einerseits derart konkret zu verfassen, dass die durch den Technikeinsatz betroffenen sozialen Funktionen ersichtlich werden, allerdings auch abstrakt genug, um die technische Gestaltung nicht vorwegzunehmen. In beiden Fällen geht es darum dazu passende Formulierungen zu finden. Diese lassen sich nicht allgemeingültig festlegen, sondern müssen entsprechend der Begleitumstände ausgearbeitet werden. So ist es beispielsweise bei der Entwicklung einer Smartphone App nicht sinnvoll im Szenario auf den Begriff „Handy“ oder „Benachrichtigungstonsignal“ zu verzichten. Es

wäre aber besser vom „Verschicken einer verschlüsselten Nachricht“ anstatt vom „Schreiben einer Email mit SSL-Verschlüsselung“ zu sprechen, weil dadurch eine technische Lösung festlegt wird zu der es Alternativen gibt.

2.5.2 Aktivität: Anwendungsszenario erarbeiten

Die Szenarien-Entwicklung erfolgt iterativ. Nach einer gemeinsamen Festlegung auf eine bestimmte Anzahl von Szenarien sowie das zu erreichende Ziel pro Szenario, erfolgt ein erster Entwurf durch den Usability Engineer. Darauf folgend wird eine gemeinsame Abstimmung der Szenarien durchgeführt, in welcher die verschiedenen Disziplinen Verbesserungsvorschläge einbringen. Dabei müssen besonders der Jurist und der Trust Engineer darauf achten, dass die Szenarien nicht derart grob und vereinfacht beschrieben sind, dass keine Aussagen bezüglich dieser zwei Disziplinen möglich sind. Eventuell müssen Details, wie das Senden der Nachricht in verschlüsselter Form, nachträglich hinzugefügt werden. Verbesserungsvorschläge der verschiedenen Disziplinen werden nach Zustimmung durch alle Disziplinen anschließend vom Usability Engineer eingearbeitet. Es folgt die abschließende Verabschiedung der Szenarien. Szenarien und Personas werden in der Aktivität „Personas und Szenarien validieren“ mit den potentiellen Nutzern validiert und wieder optimiert.

Die Anzahl der Szenarien richtet sich nach dem Umfang und der Komplexität des geplanten Systems sowie der Größe, den Zielen und Bedürfnissen der Personas. Eventuell können Szenarien gruppiert und priorisiert werden. Zur Priorisierung können etwa die Relevanz des Szenarios und die Auftrittshäufigkeit zur Laufzeit geschätzt werden. Besonders relevante Szenarios umfassen Kernaufgaben und die Hauptpersona und sind damit für den weiteren Erfolg von zentraler Bedeutung.

2.5.3 Aktivität: Geschäftsmodell erarbeiten

Das Geschäftsmodell dient der Überprüfung der Marktfähigkeit der Anwendung. Als Erweiterung der Szenarien sollten im Geschäftsmodell Informationen enthalten sein, die für eine Gestaltung der Technik notwendig sind. Das Geschäftsmodell enthält eine Definition des Nutzenversprechens [43] und beschreibt dabei Anbieter und Kunden sowie den Nutzen, den der Kunde von der Anwendung hat. Das Geschäftsmodell beschreibt zudem die Wertschöpfung (Leistungsgenerierung) und das Ertragsmodell. Somit enthält das Geschäftsmodell die wirtschaftlich relevanten Angaben. Im Sinne der Rechtmäßigkeit und Rechtsverträglichkeit (vgl. Abschnitt 3.1) ist hierbei ein Geschäftsmodell anzustreben, dessen Umsetzung keine Rechtsverletzungen nach sich zieht oder sogar rechtliche Vorgaben optimal berücksichtig [44]. Das Geschäftsmodell veranschaulicht dabei in vereinfachter Form, welche Ressourcen zur Verfügung stehen und wie diese in vermarktungsfähige Informationen, Produkte, Dienstleistungen oder hybride Produkte [45] transformiert werden sollen [46]. Dabei veranschaulicht es die Mittel und Wege, wie die Anwendungsidee erfolgreich eingesetzt werden soll.

2.5.4 Aktivität: Szenarien überprüfen

Die Persona- und Szenarien-Validierung ist eine etablierte Methode des Requirement-Engineering und wird dort für verschiedene Aktivitäten im Anforderungsanalyse-Prozess eingesetzt [37] Die Validierung der Szenarien und der Persona-Steckbriefe wird mit potentiellen Anwendern des Systems durchgeführt. Das Ziel dieser Rückkopplung mit den zukünftigen Nutzern ist die qualitative Überprüfung und Verbesserung sowohl der Szenarien als auch der Persona-Beschreibungen hinsichtlich ihrer Sinnhaftigkeit, Glaubwürdigkeit und Nachvollziehbarkeit. Zum einen können fälschliche Vermutungen über die Persona und die Verwendung des Systems durch die Validierung

aufgedeckt werden. Des Weiteren werden die Teilnehmer der Validierung bezüglich Ihrer Akzeptanz für die Szenarien befragt. Die sich ergebenden Defizite und Änderungsvorschläge werden den mitwirkenden Rollen der Vorgehensbausteine „Personas erarbeiten“ und „Anwendungsszenarien erarbeiten“ übergeben, damit diese die erforderlichen Korrekturmaßnahmen durchführen können. Unter Umständen werden weitere Nutzerbedürfnisse aufgedeckt, die in Nutzungsanforderungen zum Zwecke einer höheren Nutzerakzeptanz umgesetzt werden können.

Szenarien eignen sich für die Einbeziehung der zukünftigen Nutzergruppe, da sie konkrete Beispiele für die intendierte Systemverwendung dokumentieren [37]. Die Validierung der Szenarien ist qualitativer Art und sollte in Anlehnung an die aufwandsgünstige heuristische Evaluation von Jakob Nielsen [47] mit mindestens fünf potentiellen Nutzern durchgeführt werden. Der Einsatz von mehr als fünf Nutzern resultiert bei der Durchführung der heuristischen Evaluation zu einer unwesentlich erhöhten Aufdeckung weiterer Änderungsmaßnahmen [47], was den dafür notwendigen Aufwand i.d.R. nicht rechtfertigt. Es wird angenommen, dass die Testpersonen als Experten des zukünftigen Systems zu verstehen sind. Essentiell hierbei ist, dass die eingesetzten Testpersonen auch zur Nutzergruppe gehören, was mit Hilfe gezielter Fragen in der Validierung abgeklärt werden muss. Eine Validierung durch das eigene Projektteam ist nicht zu empfehlen, insbesondere nicht durch die Entwickler, da diese bereits eine bestimmte Betriebsblindheit aufweisen können.

Gemäß des Vorgehensbausteins wird jedes Szenario durch die Ziele einer Persona initiiert. Ein Szenario beschreibt exemplarische Abläufe zur Erfüllung dieser Persona-Ziele sowie Abläufe, die beim Eintreten definierter Ausnahmesituationen ausgeführt werden. Auf folgende Fragen wird den Entwicklern mittels der Persona- und Szenarien-Validierung eine nutzerperspektivische Antwort gegeben:

- Sind die „richtigen“ Personas für das System gewählt worden?
- Sind die „richtigen“ Verwendungsziele (Szenarien) gewählt worden?
- Sind den Personas die „richtigen“ Ziele als Systemverwendungsgrund zugeschrieben worden? (Kombination von Persona und Szenario)
- Sind die „richtigen“ Abläufe für das Erreichen eines Ziels beschrieben worden?
- Wird das Szenario von den Nutzern akzeptiert?

Dabei bedeutet „richtig“, „[…] dass ein System entwickelt wird, das den konsolidierten Wünschen und Vorstellungen der Stakeholder genügt und dabei alle Rahmenbedingungen enthält.“ [37, p. 169]. Infolge der Validierungsergebnisse erfahren die Persona-Steckbriefe, die individuellen Persona-Ziele und die Nutzungsabläufe in den Szenarien eine Korrektur und Ergänzung. Gegebenenfalls werden auch Anwendungsziele konkretisiert oder neue Anwendungsziele identifiziert.

2.5.5 Aktivität: Nutzerakzeptanz evaluieren

Die Szenario-Beschreibungen werden den Testpersonen vorgelegt, die als Nutzer in Frage kommen. Die Beschreibungen werden hinsichtlich der allgemeinen Qualität und der persönlichen Akzeptanz bewertet. Zu diesem Zweck eignet sich zunächst die Erstellung zweier Matrizen, in welcher die Szenarien in den Zeilen und die Testpersonen in den Spalten gelistet werden. In der ersten Matrix sollen die Testpersonen mittels einer Punkteskala von 0 bis 5 zum Ausdruck bringen, für wie gut sie die Szenarien-Beschreibungen im Hinblick auf Relevanz und Realitätsnähe halten. Dabei bezeichnen 5 Punkte die höchste Relevanz und Realitätsnähe, 0 Punkte dagegen eine negative Bewertung. Tabelle 1 gibt ein Beispiel. Die letzte Spalte zeigt die Medianwerte an.

Testperson / Szenario	1	2	3	4	5	Median
Szenario 1: Alice lädt mittels Meet-U Freunde zum Kino ein	4	3	4	5	4	4
Szenario 2: Alice sucht mit Meet-U nach Sport-Events in ihrer Nähe	1	2	3	2	2	2
Szenario 3: …	…	…	…	…	…	…

Tabelle 1: Einschätzung von Relevanz und Realitätsnähe der Szenarien durch Testpersonen

Die zweite Matrix ist formal identisch zur ersten Matrix aus Tabelle 1. Hier drücken die Testpersonen allerdings ihre persönliche Akzeptanz für die Szenarien aus. Es geht hierbei darum zu erfahren, ob die Testpersonen sich vorstellen können die Anwendung wie in den Szenarien beschrieben tatsächlich selbst zu nutzen. Eine 5 bedeutet die höchste und die 0 die niedrigste Bereitschaft. Dabei müssen den Testpersonen im Voraus kurz diejenigen Aspekte vorgestellt werden, auf welche gleichfalls Acht gegeben werden soll. Hierzu zählen insbesondere der Automatisierungsgrad der Anwendung bezogen auf Datenverarbeitung und Entscheidungsprozess, das Kontrollbedürfnis des Nutzers und die eventuelle Verarbeitung und Sammlung der persönlichen Daten beim Anbieter. Die Testpersonen müssen zusätzlich darauf aufmerksam gemacht werden, dass zu den persönlichen Daten auch Kontextdaten zählen.

Bewertet ein Nutzer in einem der Matrizen ein Szenario mit weniger als 4 Punkten, wird der Nutzer gebeten, die Gründe für die Ablehnung in einem Freitext zu formulieren. Die Ergebnisse zur Nutzerakzeptanz werden ausgewertet und Änderungsmöglichkeiten für den weiteren Entwicklungsprozess ausgearbeitet. Szenarien, welche in einem der Matrizen im Schnitt mit weniger als 4 Punkten bewertet worden sind, müssen mit den mitwirkenden Rollen des Vorgehensbausteins „Anwendungsszenarien erarbeiten“ überarbeitet werden. Dabei werden die Freitexte der Testpersonen gemeinsam ausgewertet.

2.5.6 Aktivität: Personas und Szenarien validieren

Ziel dieser Teilaktivität ist die Aufdeckung von Problemen im Hinblick auf die Sinnhaftigkeit, Glaubwürdigkeit und die Nachvollziehbarkeit der Personas und Szenarien.

Dazu werden den Testpersonen nacheinander die einzelnen Szenarien und die jeweils integrierten Personas anhand deren Steckbriefe vorgelegt. Die Testpersonen lesen einzeln die Szenarien. Mit Hilfe einer Leitfragen-Übersicht werden dann folgende Validierungsfragen gestellt und die Antworten der Testpersonen im Leitfaden notiert:

- Wie realistisch sind die Personas und Szenarien jeweils?
- Würden die angedachten Personas das System verwenden?
- Hilft das in den Szenarien dargestellte System, dass die Personas ihre Ziele erreichen?
- Bewerten Sie den im Szenario dargestellten Interaktionsablauf!
- Gibt es weitere (Teil-)Szenarien, die das System leisten sollte?

Jede Antwort erfordert eine Begründung. Auf diese Weise werden Hinweise zur Validität und zu Optimierungsmaßnahmen geliefert. Bei der Validierung ist es wichtig, dass die von den Testpersonen geäußerten Antworten, die unter Umständen unklar und unspezifisch sein können, geklärt werden. Des Weiteren müssen die Fragen auf eine Weise gestellt werden, welche keine vorgefertigten

Antworten in den Mund legt. Die handschriftlich notierten Antworten aller Testpersonen je Szenario und Persona-Steckbrief werden vom Usability Engineer pro Szenario und Persona strukturiert zusammengefasst. Anschließend werden die Ergebnisse allen Mitwirkenden vorgelegt, um eine darauf folgende Überarbeitung zu ermöglichen. Dabei werden insbesondere mehrfach vorkommende Kritik und Vorschläge aufgegriffen und diskutiert. Verbesserungsvorschläge werden abgestimmt. Die Einarbeitung der Optimierungsvorschläge erfolgt durch den Usability Engineer.

2.5.7 Aktivität: Überprüfen der technischen Realisierbarkeit

Zur Überprüfung der technischen Realisierbarkeit müssen zunächst zentrale Leistungsmerkmale aus den Projektideen extrahiert werden. Für jedes Leistungsmerkmal ist nun zu überprüfen, ob die aktuell vorhanden technischen Mittel, oder eine technische Aufrüstung, ausreichen, um das jeweilige Merkmal zu erfüllen. Die folgenden Fragen sind Hilfsmittel, um die technischen Realisierbarkeit zu beurteilen.

- Ist die aktuell auf dem Markt vorhandene Technologie geeignet, um die Leistungsmerkmale zu erfüllen?
- Sind die dem Projekt zur Verfügung stehenden finanziellen Mittel ausreichend, um die Leistungsmerkmale zu erfüllen?
    - Ist genügend Personal vorhanden?
    - Ist geeignete Software/Hardware vorhanden?
    - Ist die technische Expertise vorhanden?
    - Ist der Zeitplan angemessen?
- Können Leistungsmerkmale angepasst werden (Alternativen), so dass diese mit den vorhandenen Mitteln erfüllbar sind?

Reichen die aktuell verfügbaren Mittel nicht aus um alle Leistungsmerkmale adäquat zu erfüllen. müssen Alternativvorschläge erarbeitet werden. Bildet keiner der Alternativvorschläge eine realisierbare Lösung, so ist die in den Anwendungsszenarien konkretisierte Projektidee nicht durchführbar.

Beispiel Anwendungsszenario für Meet-U: „Event suchen“

Alice möchte mit Freunden am Samstagabend ins Kino gehen und möchte für die Planung die neue App Meet-U verwenden.

Meet-U verlangt von ihr bei der erstmaligen Nutzung das Angeben persönlicher Daten wie z. B. ihren Vor- und Zunamen, ihre E-Mail-Adresse und ihre Interessen und Vorlieben etwa im Hinblick auf Events, die von ihr bevorzugten Verkehrsmittel, ihre Lieblingsmusik oder ihr von bevorzugten Filmgenres. Diese Angaben werden im Rahmen ihres persönlichen Profils gespeichert. Darüber hinaus fügt sie noch ein paar Freunde zu ihrer Kontaktliste hinzu.

Für die Samstagsabendplanung sucht sie einen Film, der sie interessiert, ein Kino, das ihn zeigt aus den Vorschlägen von Meet-U aus und wählt die Personen aus ihren Kontakten, die motiviert sein könnten mitzukommen und die sie gerne dabei hätte. Außerdem wählt sie einen Treffpunkt am Kino, an dem alle rechtzeitig zusammen kommen sollen. Nach dem Verschicken der Einladung unterstützt Meet-U bei der Platzreservierung im Kino. Die Anwendung informiert Alice auch, ob ein Freund dem Treffen zugesagt oder abgesagt hat.

Während Alice gerade dabei ist ihr Zimmer aufzuräumen, erinnert Meet-U sie daran, dass sie aufgrund ihrer aktuellen Entfernung zum Kino und der Uhrzeit in den nächsten 30 Minuten aufbrechen sollte, um pünktlich beim Treffen zu sein. Zudem wird sie von Meet-U bei Ereignissen informiert, die für sie nach ihren Voreinstellungen relevant sind, wie z. B. wenn alle ihre Freunde zugesagt haben oder der Kinofilm ausfällt. Meet-U zieht dabei nur wenn nötig Aufmerksamkeit auf sich bzw. hatte Alice eingestellt, bei welcher Art Neuigkeiten bezüglich des Events sie sofort informiert werden möchte.

Alice lässt sich von Meet-U zum Kino navigieren, weil sie den Weg dorthin nicht kennt. Währenddessen kann Alice sehen, ob sich die eingeladenen Freunde auch auf dem Weg zum Kino befinden. Leider hat Peters Auto einen platten Reifen, weshalb er mit einer kurzen Begründung kurzfristig absagen muss.

Am Kino angekommen wartet sie an dem mit Meet-U vereinbarten Treffpunkt auf ihre Freunde. Das Kino bietet einen Ticketservice und einen Gebäudeplanservice an. Alice bezahlt ihr Ticket über den Ticketservice und druckt ihr Ticket an einem dafür vorgesehenen Automaten aus.

Nachdem ihre Freunde eingetroffen sind und auch ihre Karten erworben haben, gehen sie gemeinsam in den Vorführungssaal. Meet-U schaltet daraufhin die Smartphones stumm, damit sie während des Films nicht stören können.

3 Anforderungsmanagement (Phase 2)

Besonderheiten der Anforderungserhebung in der VENUS-Entwicklungsmethode:

- Normative Anforderungserhebung für die soziotechnischen Anforderungen der Bereiche Rechtsverträglichkeit, Vertrauenswürdigkeit, Gebrauchstauglichkeit und Softwarequalität
- Interdisziplinäre Anforderungsvereinbarung mit GSS
- Nutzung standardisierte Anforderungen (Anforderungsmuster) der Rechtsverträglichkeit, Vertrauenswürdigkeit und Softwarequalität

Die Phase des Anforderungsmanagement der VENUS-Entwicklungsmethode bringt Hilfestellung für die interdisziplinäre Anforderungserhebung (für Anforderungen der üblichen Stakeholder sollten Standardverfahren genutzt werden). Der Trust Engineer, der Usability Engineer, der Jurist und der System Engineer nutzen jeweils eigene Vorgehensbausteine, um technische Anforderungen zu erarbeiten. Diese Vorgehensbausteine beruhen dabei auf vergleichbaren Vorgehensweisen zur Konkretisierung normativer Vorgaben (siehe Abbildung 3). Die Identifizierung und Konkretisierung normativer Vorgaben und Kriterien sowie technischer Anforderungen erfolgt in drei Stufen. Die Stufen werden grundsätzlich nacheinander abgearbeitet, es kann aber ergänzend ein iteratives Vorgehen erforderlich sein, z. B. um Korrekturen vorzunehmen.

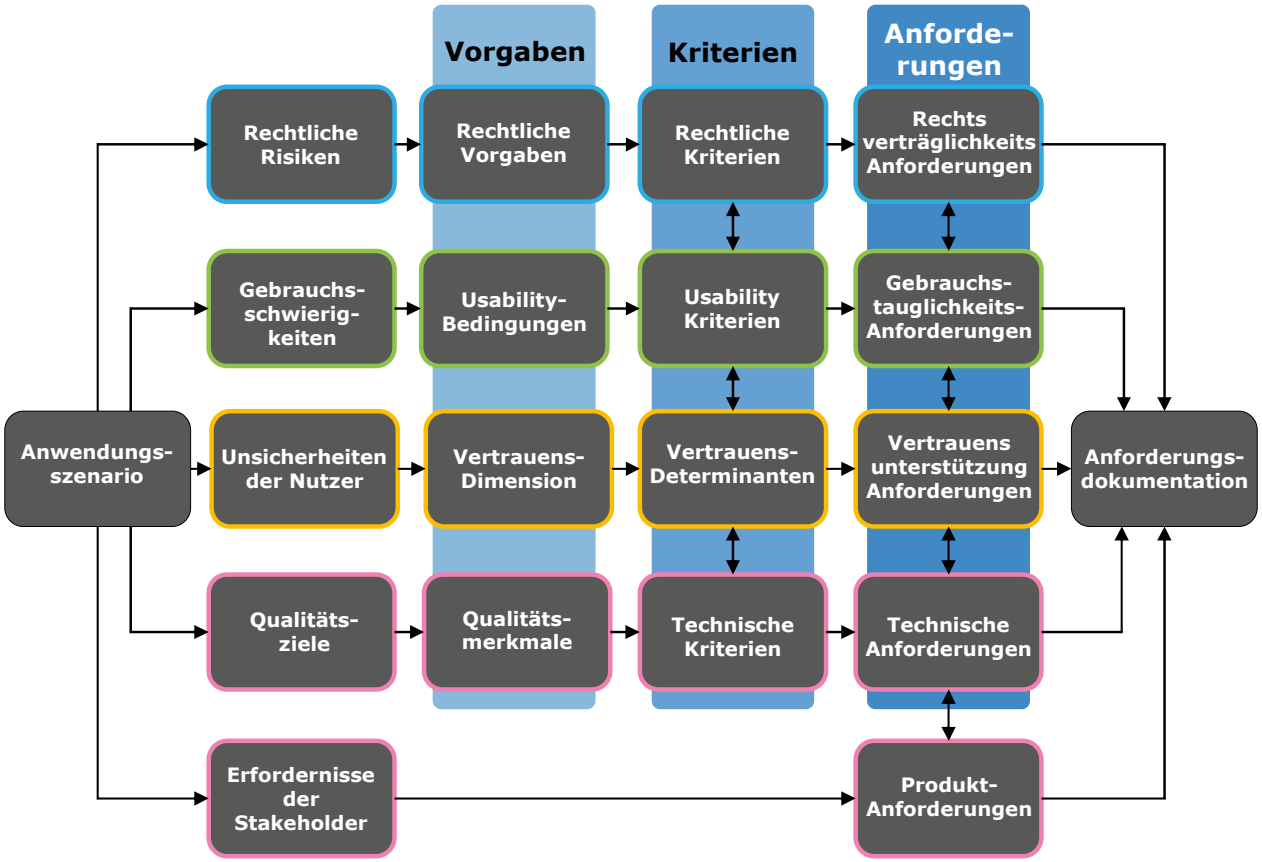


Abbildung 3: Anforderungserhebung in der VENUS-Entwicklungsmethode

Die einzelnen Disziplinen formulieren normative Vorgaben, die sich jeweils aus den Fachdisziplinen bezogen auf die konkret zu gestaltende Technik ergeben. Normative Vorgaben sind Regeln, die etwas festlegen und die einzuhalten sind. Jede Fachdisziplin verfügt über allgemeine Vorgaben, die stets bei

bestimmten Technikklassen, wie zum Beispiel Kommunikationstechnik, zu berücksichtigen sind. Ergänzend können sich aus jeder Fachdisziplin technik- und anwendungsspezifische Vorgaben in Bezug auf die konkrete Technik und ihre Anwendungsbereiche ergeben.

Aus den heterogenen Vorgaben werden in der zweiten Stufe normative Kriterien ermittelt. Normative Kriterien sind die konkretisierten normativen Vorgaben, die jedoch noch nicht so konkret sind, dass sie unmittelbar technisch umgesetzt werden können. Dabei soll das gesamte normative Spektrum abgedeckt werden. Normative Kriterien werden erarbeitet, indem nach Regeln gesucht wird, die die normativen Vorgaben erfüllen können, die sich durch die Technik ergeben.

Aus den normativen Kriterien werden auf der dritten Stufe technische Gestaltungsanforderungen formuliert. Technische Gestaltungsanforderungen sind Anforderungen, die eine Aussage über die Beschaffenheit oder Fähigkeit eines Systems treffen. Diese werden gewonnen, indem von der vorhandenen Technik ausgehend, die Gestaltung der Technik so gewählt wird, dass die normativen Kriterien erfüllt werden. In diesem Schritt erfolgt ein Wechsel von einer normativen zu einer technischen Sprache. Dadurch können einheitliche technische Ziele im Rahmen einer sozialverträglichen Informationstechnikgestaltung erreicht werden.

Am Ende werden die Anforderungen aus den Vorgehensbausteinen gemeinsam vereinbart und ein konsolidiertes Anforderungsdokument für die weitere Entwicklung erstellt. Die Erhebung der fachspezifischen Anforderungen kann durch den Einsatz von Anforderungsmustern erleichtert werden. Dies ist abschließend als Werkzeugunterstützung zu dieser Phase beschrieben.

3.1 VB: Normative Anforderungserhebung Rechtsverträglichkeit

Besonderheiten der Anforderungserhebung Recht in der VENUS-Entwicklungsmethode:

- Gestaltung der Applikationen aus rechtlicher Sicht von Beginn an
- Ableitung von Anforderungen auch ohne detaillierte Detailgesetze aus den höheren Gesetzesebenen (somit Eignung für neuartige Systeme)
- Ziel der optimalen Berücksichtigung der sozialen Regelungsziele des Rechts (Rechtsverträglichkeit)

Produkt	Rechtliche Anforderungen
Verantwortliche Rolle	Jurist
Mitwirkende Rollen	Anforderungsanalyst

Aus rechtlicher Sicht besteht die Schwierigkeit, dass Rechtsnormen selten konkrete Vorgaben für die Gestaltung von UC-Anwendungen enthalten. Das gilt insbesondere für Grundrechte, da sie sehr allgemein formuliert sind. Sie enthalten vor allem Regeln für das Zusammenleben der Menschen und beschreiben nicht den Gebrauch bestimmter UC-Anwendungen. Da aber der Mensch und die Kommunikation zwischen Menschen durch UC-Technik allgegenwärtig unterstützt werden, müssen sich die rechtlichen Vorgaben auch auf die Technik auswirken. Daher müssen Entwickler derartiger Anwendungen rechtliche Vorgaben beachten und in technische Anforderungen umsetzen. Der erforderliche Vermittlungsschritt zwischen Recht und Technik wird durch die Methode KORA (Konkretisierung rechtlicher Anforderungen; siehe zu der Methode etwa [48], [49, p. 46 f.], [50]) abgebildet. Mit Hilfe von KORA ist es möglich, technische Anforderungen, die eine möglichst optimale Sicherung rechtlicher Ziele gewährleisten, zu erarbeiten. Zunächst müssen allerdings die (Grund-)Rechte identifiziert werden, deren Verwirklichungsbedingungen durch den Einsatz der zu entwickelnden UC-Anwendung beeinflusst werden. Die Methode KORA umfasst grundsätzlich vier Stufen. Für die Anforderungserhebung, also die Ableitung von technischen Anforderungen aus rechtlichen Vorgaben, sind aber zunächst nur die in Abbildung 4 bezeichneten ersten drei Stufen relevant. Nachdem zunächst abstrakte (Grund-)Rechte identifiziert wurden (Vorstufe), werden aus diesen rechtliche Vorgaben (1. Stufe) gewonnen. Daraus können dann rechtliche Kriterien (2. Stufe) abgeleitet werden, die wiederum zu technischen Anforderungen (3. Stufe) konkretisiert werden.

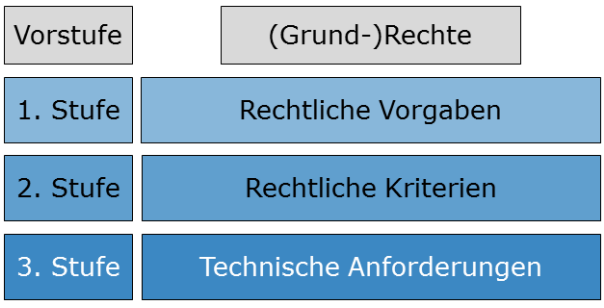


Abbildung 4: KORA, Stufen 1-3

Bei der Erarbeitung der Vorgaben, Kriterien und Anforderungen ist nicht stets eine Stufe nach der anderen zu erarbeiten. Es handelt sich vielmehr um einen iterativen Prozess. Ziel ist ein schlüssiges Gesamtbild rechtlicher Vorgaben, rechtlicher Kriterien und technischer Anforderungen. Werden zunächst einzelne Kriterien oder Anforderungen ermittelt, ist zu prüfen, welche übergeordneten

Kriterien oder Vorgaben durch diese konkretisiert werden. Die einzelnen Stufen werden, nach der Beschreibung des Produktes, näher erläutert.

3.1.1 Produkt: Rechtliche Anforderungen

Die gesammelten rechtlichen Vorgaben, rechtlichen Kriterien und technischen Anforderungen werden dokumentiert. Der Inhalt der einzelnen Vorgaben, Kriterien und Anforderungen wird dabei textlich beschrieben. Außerdem werden die Beziehungen zwischen den Vorgaben, Kriterien und Anforderungen verdeutlicht. Es wird veranschaulicht, welche Kriterien welche Vorgaben und welche Anforderungen welche Kriterien konkretisieren, so dass auch später noch nachvollzogen werden kann, welche Anforderungen welche Kriterien und Vorgaben umsetzen. So sind, falls Änderungen erforderlich werden, die Konsequenzen sofort erkennbar und das Gesamtbild der technischen Anforderungen bleibt überprüfbar.

3.1.2 Aktivität: Identifizierung relevanter (Grund-)Rechte

Die rechtliche Mitgestaltung von UC-Anwendungen ist darauf gerichtet, die zu erwartenden rechtlich relevanten Auswirkungen zu beeinflussen. Dabei ist der Blick sowohl auf die Chancen, also die als vorteilhaft anzusehenden Auswirkungen, als auch auf die als nachteilig zu bewertenden Auswirkungen, also die Risiken zu richten. Im Rahmen einer Chancen- und Risikenanalyse wird zum einen untersucht, wie sich durch die zu entwickelnde Technik die Verwirklichungsbedingungen von (Grund-)Rechten verbessern werden oder verbessern könnten. Zum anderen werden aber auch drohende Beschränkungen für die Wahrnehmung von (Grund-)Rechten analysiert. Grundlage für diese Untersuchung sind die bereits während der Bedarfsanalyse entwickelten Szenarien sowie das Geschäftsmodell.

Typisch für UC-Anwendungen ist etwa der massenhafte Umgang mit personenbezogenen Daten. Vor diesem Hintergrund wird im Zusammenhang mit UC-Anwendungen das allgemeine Persönlichkeitsrecht nach Art. 2 Abs. 1 GG i. V. m. Art. 1 Abs. 1 GG häufig eine zentrale Rolle spielen. Welche anderen (Grund-)Rechte betroffen sind, hängt wesentlich vom Einzelfall, insbesondere von der jeweils zu entwickelnden Anwendung, dem Einsatzbereich und den potentiellen Nutzern ab. Werden UC-Anwendungen etwa für ein berufliches Umfeld entwickelt, ist beispielsweise zusätzlich das Grundrecht der Berufsfreiheit nach Art. 12 GG zu berücksichtigen.

3.1.3 Aktivität: Identifizierung rechtlicher Vorgaben

Als Ausgangspunkt für die Gewinnung der rechtlichen Vorgaben dienen die zuvor identifizierten (Grund-)Rechte, die für die rechtliche Betrachtung der jeweiligen UC-Anwendung relevant sind. Die rechtlichen Vorgaben werden dadurch gewonnen, dass die identifizierten (Grund-)Rechte bezogen auf die von der Anwendung betroffenen sozialen Funktionen rechtlich interpretiert werden. So ergeben sich rechtliche Vorgaben, die für die Anwendung umzusetzen sind. Gegenstand rechtlicher Vorgaben sind also keine Merkmale der Technik, sondern rein soziale Funktionen, die aber durch Technik erbracht oder verändert werden können. Für die Gewinnung der rechtlichen Vorgaben können typische rechtliche Auslegungsmethoden (siehe dazu [51, p. 312 ff.]) verwendet werden. Häufig kann auch auf Rechtsprechung zurückgegriffen werden. Dies ist beispielsweise beim Recht auf informationelle Selbstbestimmung oder beim Grundrecht auf Gewährleistung der Vertraulichkeit und Integrität informationstechnischer Systeme der Fall, welche das Bundesverfassungsgericht aus dem allgemeinen Persönlichkeitsrecht hergeleitet hat [52], [53]. Rechtliche Vorgaben können etwa die

informationelle Selbstbestimmung (Schutz der Entscheidungsfreiheit des Einzelnen über die Preisgabe und Verwendung seiner persönlichen Daten und ob und inwieweit von Dritten über seine Persönlichkeit mit dieser Verarbeitung verfügt werden kann), die kommunikative Selbstbestimmung (Schutz des aktuellen Prozesses der Identitätsbildung und Selbstdarstellung) oder die Entfaltungsfreiheit (Möglichkeit sich frei von Zwängen zu betätigen) sein.

3.1.4 Aktivität: Identifizierung rechtlicher Kriterien

Auf der zweiten Stufe werden die so gebildeten rechtlichen Vorgaben zu rechtlichen Kriterien konkretisiert. Die rechtlichen Kriterien werden gewonnen, indem nach Regeln gesucht wird, mit denen die rechtlichen Vorgaben vor dem Hintergrund der spezifischen Eigenschaften, Risiken und Anwendungsbedingungen der Technik erfüllt werden können [49]. Die Kriterien enthalten sowohl Bezüge zu technischen Funktionen als auch zu den rechtlichen und sozialen Aspekten. Sie beschreiben relativ abstrakte Problemlösungen in Bezug auf die Vorgaben, weisen jedoch noch keinen Bezug zu einem bestimmten technischen, organisatorischen oder rechtlichen Lösungsansatz auf. Rechtliche Kriterien können zum Beispiel auch aus den implizit im einfachen Recht enthaltenen Maßstäben oder aus Gründen, die von Richtern in solchen Rechtsfällen zur Urteilsfindung herangezogen wurden, in denen die gleichen Vorgaben eine Rolle spielten, gewonnen werden. Rechtliche Kriterien, die eine Konkretisierung der informationellen und der kommunikativen Selbstbestimmung darstellen, können etwa Zweckfestlegung und Zweckbindung (Verarbeitung personenbezogener Daten nur zur Erfüllung der zuvor festgelegten Zwecke), Erforderlichkeit (Begrenzung der Anwendung derart, dass die Beeinträchtigungen verbürgter Rechte auf das unbedingt erforderliche Maß begrenzt werden) und Datenvermeidung und Datensparsamkeit (möglichst weitgehende Reduzierung der Verarbeitung und Weitergabe personenbezogener oder sonstiger vertraulich zu behandelnder Daten) sein. Das Kriterium der Datenvermeidung und Datensparsamkeit setzt auch die Vorgabe der Entfaltungsfreiheit um.

3.1.5 Aktivität: Identifizierung technischer Anforderungen

Bei den technischen Anforderungen handelt es sich um abstrakte technische Gestaltungsziele, die verfolgt werden sollten, um die herausgearbeiteten Kriterien zu erfüllen. Gewonnen werden sie dadurch, dass von der Technik her gefragt wird, wie die technische Anwendung zu gestalten ist, um die identifizierten rechtlichen Kriterien zu erfüllen [49]. Technische Anforderungen sind auf einem relativ hohen Abstraktionsniveau formuliert und stellen noch keine konkreten Technikmerkmale dar. Sie können etwa Grundfunktionen einer UC-Anwendung sein oder organisatorische Gesichtspunkte beachten. Da KORA nicht nur eine rechtmäßige, sondern darüber hinausgehend eine rechtsverträgliche Gestaltung der Anwendungen anstrebt, stellt sich bei den technischen Anforderungen die Frage nach einer Gestaltung der Technik, welche die Rechtsverträglichkeit der UC-Anwendung über ein Maß der Rechtskonformität hinaus erhöhen kann. Ziel ist die bestmögliche Verwirklichung der rechtlichen Vorgaben und Kriterien. Dabei sollte beachtet werden, dass es sich bei den technischen Anforderungen sowohl um anzustrebende „Soll-Anforderungen“, als auch um „Muss-Anforderungen“ handeln kann. Aus den Vorgaben informationelle Selbstbestimmung, kommunikative Selbstbestimmung und Entfaltungsfreiheit kann sich in Konkretisierung der Kriterien Zweckfestlegung und Zweckbindung, Erforderlichkeit sowie Datenvermeidung und Datensparsamkeit beispielsweise ergeben, dass Daten, die nicht mehr benötigt werden, automatisch zu löschen sind.

Beispiel Anforderungen der Rechtsverträglichkeit für Meet-U

Bei der Erarbeitung der technischen Anforderungen für Meet-U wurde ein Anforderungserhebungskatalog erstellt, der wie folgt aussieht:

Rechtliche Vorgaben

- V1: Informationelle Selbstbestimmung (Art. 2 Abs. 1 i. V. m. Art 1 Abs. 1 GG): Die Anwendung muss die Entscheidungsfreiheit des Einzelnen über die Preisgabe und Verwendung seiner persönlichen Daten und ob und inwieweit von Dritten über seine Persönlichkeit mit dieser Verarbeitung verfügt werden kann, schützen.
- V2: Kommunikative Selbstbestimmung (Art. 2 Abs. 1 i. V. m. Art 1 Abs. 1, Art. 10 GG): Die Anwendung muss den aktuellen Prozess der Identitätsbildung und Selbstdarstellung schützen.
- V3: Entfaltungsfreiheit (Art. 2 Abs. 1 GG): Die Anwendung muss es dem Nutzer erlauben, sich frei von Zwängen zu betätigen.
- V4: …

Rechtliche Kriterien

- K1: Zweckfestlegung und Zweckbindung (V1, V3): Die Verarbeitung personenbezogener Daten ist nur insoweit zulässig, wie sie zur Erfüllung der zuvor festgelegten Zwecke unerlässlich ist.
- K2: Erforderlichkeit (V1, V3): Die Anwendung muss so gestaltet sein, dass die Beeinträchtigungen verbürgter Rechte auf das unbedingt erforderliche Maß begrenzt werden.
- K3: Datenvermeidung und Datensparsamkeit (V1, V2, V3): Die Verarbeitung personenbezogener Daten oder sonstiger vertraulich zu behandelnder Daten und die Weitergabe der Daten an Dritte muss soweit wie möglich reduziert werden.
- K4: …

Technische Anforderungen

- A1: Daten, die nicht mehr benötigt werden, sollen sofort automatisch gelöscht werden. (K1, K2, K3)
- A2: …

3.2 VB: Normative Anforderungserhebung Vertrauenswürdigkeit

Besonderheiten der Anforderungserhebung Vertrauen in der VENUS-Entwicklungsmethode:

- Berücksichtigung der Vertrauenswürdigkeit der Applikation und somit die Nutzerakzeptanz von Beginn an
- Nutzung theoretischer Erkenntnisse der Vertrauensbildung

Produkt	Vertrauensbezogene funktionale Anforderungen
Verantwortliche Rolle	Trust Engineer
Mitwirkende Rollen	Anforderungsanalyst Nutzer

Mehrere Studien haben Vertrauen als einen der zentralen Faktoren bei der Entscheidung eines Nutzers, neue Technologie zu nutzen oder nicht, identifiziert. Vertrauen ist definiert als „der Eindruck des Vertrauensgebers, dass ein bestimmter Vertrauensnehmer ihm in einer Situation, die von Unsicherheit und Verwundbarkeit des Vertrauensgebers charakterisiert ist, dabei behilflich sein wird, seine Ziele zu erreichen“ [54]. Während eine Vielzahl verhaltensorientierter Arbeiten untersucht haben, wie Vertrauen zu Stande kommt, existieren nur wenige Erkenntnisse darüber, wie die verhaltensorientierten Ergebnisse systematisch in die Entwicklung neuer technischer Systeme und Anwendungen einfließen können. Um diese Lücke zu schließen, wurde im Rahmen des VENUS-Projekts eine Methode entwickelt (siehe Abbildung 5), die es ermöglicht auf Basis der verhaltensorientierten Erkenntnisse zu Vertrauen konkrete Designelemente (vertrauensunterstützende Komponenten, VUK) für ein zu entwickelndes System abzuleiten (siehe [54] für eine detaillierte Beschreibung und Herleitung der Methode). Ziel der abgeleiteten Designelemente ist es, das Vertrauen des Nutzers in das System zu erhöhen.

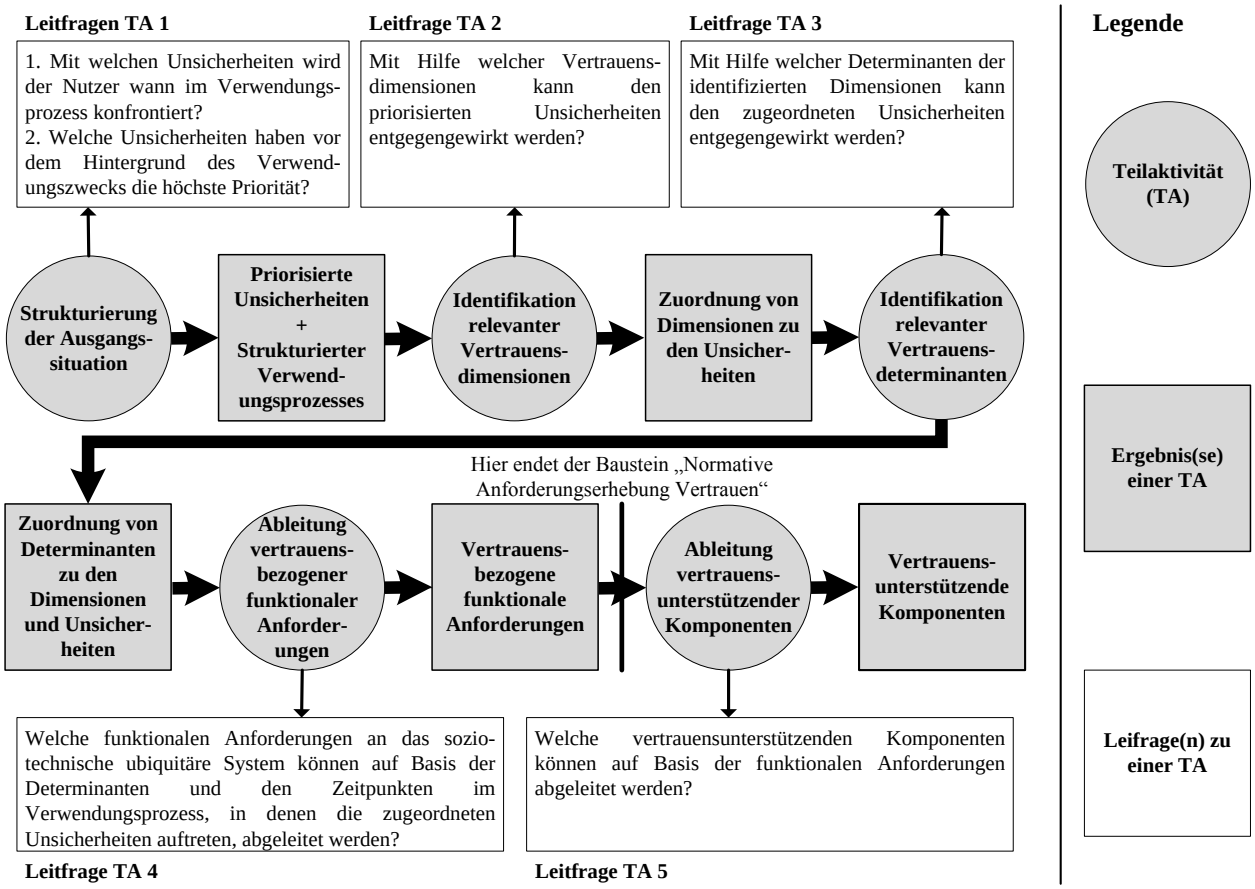


Abbildung 5: Methode zur Ableitung vertrauensunterstützender Komponenten für soziotechnische ubiquitäre Systeme (in Anlehnung an [54])

3.2.1 Produkt: Anforderungen der Vertrauenswürdigkeit

Bei der Anforderungserhebung zur Vertrauenswürdigkeit entstehen ein strukturierter Verwendungsprozess, eine priorisierte Liste der Unsicherheiten, sowie eine Übersicht, in der die relevanten Vertrauensdeterminanten und Vertrauensdimensionen den Unsicherheiten zugeordnet sind. Die Übersicht enthält zudem die konkreten vertrauensbezogenen funktionalen Anforderungen.

3.2.2 Aktivität: Strukturierung der Ausgangssituation

Frühere Studien haben gezeigt, dass Vertrauen stark von der jeweiligen Situation, in der sich der Vertrauensgeber befindet, abhängt [55]. Aufgrund der Situationsabhängigkeit müssen in der ersten Teilaktivität der Methode die Ausgangssituation des späteren Vertrauensgebers und alle relevanten Rahmenbedingungen erfasst und strukturiert werden, um die nutzerspezifischen Erwartungen und Unsicherheiten zu erkennen.

Dabei ist zu beachten, dass ein soziotechnisches ubiquitäres System (als Vertrauensnehmer) vom Nutzer als Werkzeug angesehen wird, von dessen Nutzung er Unterstützung erwartet, um ein bestimmtes Ziel zu erreichen [56], [57]. Das bedeutet, dass der Nutzer einen situationsgerechten Unterstützungsbedarf einfordert. Demnach ist im ersten Schritt der Methode der Verwendungszweck möglichst genau zu definieren, um in den folgenden Schritten Maßnahmen gezielt auf die Unterstützung des Verwendungszweckes auszurichten.

Ist der Verwendungszweck definiert, sind Zeitpunkte im Verwendungsprozess zwischen dem Nutzer und dem Dienst zu identifizieren, in denen eine Vertrauensunterstützung sinnvoll erscheint. Aus der

Vertrauenstheorie ist bekannt, dass Vertrauen nur in Situationen von Bedeutung ist, in denen Unsicherheit herrscht [58]. Eine Vertrauensunterstützung ist daher genau zu den Zeitpunkten des Verwendungsprozesses sinnvoll, in denen der Nutzer mit Unsicherheiten konfrontiert wird. Daher ist der Verwendungsprozess aus Nutzersicht zu durchlaufen und zu analysieren, wann der Nutzer mit welchen Unsicherheiten konfrontiert wird. Diese Analyse kann, abhängig von Faktoren wie Vorlaufzeit oder Expertise, vom Entwickler selbst, aus der Perspektive der Nutzer [59] oder mit potentiellen Nutzern zusammen, unter Verwendung geeigneter Methoden, geschehen. Eine geeignete Methode ist z. B. ein Interview [60] oder die Think Aloud Methode [47].

Nachdem die Unsicherheiten identifiziert wurden, mit denen der Nutzer während der Verwendung konfrontiert wird, sind diese anschließend zu priorisieren. Dies ist wichtig, da in den meisten Projekten ein gewisser Zeitdruck und begrenzte Ressourcen vorherrschen und deshalb zu erwarten ist, dass nicht alle identifizierten Unsicherheiten adressiert werden können. Daher sollten die Unsicherheiten mit der höchsten Priorität zuerst adressiert werden, da diese für den Erfolg des Systems am kritischsten sind. Die Kritikalität ist hierbei stark im Kontext des Einsatzzweckes zu sehen. Bei einem Navigationsdienst wäre eine Unsicherheit bzgl. der Richtigkeit einer vorgeschlagenen Route z. B. als deutlich kritischer anzusehen, als eine Unsicherheit bzgl. der Richtigkeit der Berechnung des zu erwartenden Benzinverbrauchs. Dies lässt sich damit begründen, dass das Auffinden der richtigen Route beim Navigationssystem Hauptzweck der Nutzung ist [54].

Ergebnisse der ersten Teilaktivität sind eine strukturierte Beschreibung des Verwendungsprozesses, durch die später nachvollzogen werden kann, wann der Nutzer mit welchen Unsicherheiten konfrontiert wird und eine priorisierte Liste der Unsicherheiten. Zusätzlich sollte vor dem Hintergrund der vorliegenden Rahmenbedingungen – z. B. verfügbare Ressourcen – definiert werden, für wie viele der Unsicherheiten vertrauensunterstützende Komponenten entwickelt werden sollen.

3.2.3 Aktivität: Identifikation relevanter Vertrauensdimensionen

Mithilfe der in der ersten Teilaktivität ermittelten und priorisierten Liste der Unsicherheiten im Verwendungsprozess werden in der zweiten Teilaktivität der Methode die relevanten Vertrauensdimensionen identifiziert, um den wichtigsten Unsicherheiten entgegenzuwirken. Dieser Schritt ist notwendig, da Vertrauen in der Literatur als mehrdimensionales Konzept und als sehr facettenreich angesehen wird [61], [57], [55]. Auch ist eine Vertrauensunterstützung auf Ebene der einzelnen Facetten systematischer möglich [54].

Vertrauen tritt in unterschiedlichen Kontexten auf [62]. Daraus ergeben sich z. B. unterschiedliche Arten von Vertrauensbeziehungen, die untersucht werden (z. B. zwischenmenschlich und zwischen Mensch und IT). Diese Tatsache macht es notwendig, dass die Identifikation möglicherweise relevanter Vertrauensdimensionen vor dem Hintergrund der vorliegenden Vertrauensbeziehung geschieht. Liegt eine zwischenmenschliche Vertrauensbeziehung vor, so sind z. B. die Vertrauensdimensionen – Fähigkeit, Wohlwollen und Integrität – von [61] oder vergleichbare – für eine Übersicht siehe [62] – grundsätzlich geeignet, um auftretenden Unsicherheiten entgegenzuwirken. Liegt hingegen eine Vertrauensbeziehung zwischen Mensch und IT vor, so sind die Vertrauensdimensionen – Performanz, Prozessnachvollziehbarkeit und Zweckklarheit – von [57] heranzuziehen, um den identifizierten Unsicherheiten im Verwendungsprozess entgegenzuwirken.

Wenn die in Betracht kommenden Vertrauensdimensionen identifiziert wurden, ist für die priorisierten Unsicherheiten jeweils die Vertrauensdimension zu identifizieren, die sich am besten zur

Vermeidung der jeweiligen Unsicherheiten eignet. Im oben genannten Beispiel des Navigationsdienstes kann der Unsicherheit bzgl. der Richtigkeit der vorgeschlagenen Route am besten mit der Vertrauensdimension Prozessnachvollziehbarkeit entgegengewirkt werden. Es ist anzunehmen, dass der Nutzer sich über genau einen bestimmten Routenvorschlag wundert – z. B. weil er im betreffenden Fall über Ortskenntnisse verfügt – daher ist es wichtig, dem Nutzer in diesem Fall verständlich zu machen, warum genau diese Route vorgeschlagen wurde, was unter die Vertrauensdimension Prozessnachvollziehbarkeit fällt.

Ergebnis der zweiten Teilaktivität ist eine, um die relevanten Vertrauensdimensionen erweiterte, priorisierte Liste der Unsicherheiten im Verwendungsprozess.

3.2.4 Aktivität: Identifikation relevanter Vertrauensdeterminanten

Wie in 3.2.3 dargestellt wurde, ist Vertrauen ein multidimensionales Konstrukt [61]; [57], wobei die einzelnen Dimensionen wieder durch unterschiedliche Determinanten beeinflusst werden. Aus diesem Grund werden in der dritten Teilaktivität der Methode relevante Vertrauensdeterminanten identifiziert, indem die um relevante Vertrauensdimensionen angereicherte Unsicherheitsliste noch weiter konkretisiert wird. Ziel ist es, den einzelnen Unsicherheiten möglichst genau eine Vertrauensdeterminante zuzuweisen, mit Hilfe derer dieser Unsicherheit entgegengewirkt werden kann, um somit das volle Potential der Vertrauensunterstützung auszuschöpfen.

Zur Identifikation geeigneter Determinanten kann bei einer zwischenmenschlichen Vertrauensbeziehung die Arbeit von [61] als Ausgangspunkt herangezogen werden, wenn es sich um eine Vertrauensbeziehung zwischen Mensch und IT handelt, enthalten die Arbeiten von [57]; [56] und [63] Sammlungen entsprechender Determinanten.

Im oben genannten Beispiel des Navigationsdienstes wurde die Prozessnachvollziehbarkeit als relevante Vertrauensdimension identifiziert, um der Unsicherheit bzgl. der Richtigkeit der vorgeschlagenen Route entgegenzuwirken. Die drei Determinanten Konsistenz [64], Verständlichkeit [65] und Kontrolle [66] stellen die drei Konkretisierungsoptionen dieser Dimension dar. Hier erscheint die Determinante Verständlichkeit am geeignetsten, um der Unsicherheit entgegenzuwirken. Dies ist damit zu begründen, dass der Nutzer nicht blind auf das System vertrauen möchte, sondern die Möglichkeit haben möchte, nachzuvollziehen, warum genau diese Route vom Navigationssystem vorgeschlagen wurde [65].

Es ist jedoch anzumerken, dass oftmals mehrere Vertrauensdeterminanten in Frage kommen, um einzelnen Unsicherheiten entgegenzuwirken. Obwohl es theoretisch möglich ist, einer Unsicherheit mit mehreren Determinanten entgegenzuwirken, sollte dies nur dann geschehen, wenn diese Unsicherheit als höchstkritisch identifiziert wurde. Das ist damit zu begründen, dass jede zu adressierende Determinante Auswirkung auf die Anzahl der abgeleiteten funktionalen Anforderungen im folgenden Methodenschritt und somit Einfluss auf den Umfang des späteren Design- und Implementierungsaufwands hat. Daher empfehlen die Autoren, auf Basis eigener Erfahrungen, beim Vorliegen verschiedener Optionen, z. B. auf Basis empirischer Erkenntnisse oder theoretischer Überlegungen, die Determinante mit dem höchsten Einfluss auszuwählen.

Ergebnis der dritten Teilaktivität ist eine, um die zu adressierenden Determinanten erweiterte, Liste der priorisierten Unsicherheiten im Verwendungsprozess.

3.2.5 Aktivität: Vertrauensbezogene funktionale Anforderungen ableiten

In der nächsten Teilaktivität der Methode werden aus den einzelnen identifizierten Determinanten konkrete funktionale Anforderungen an das zu entwickelnde System abgeleitet. Um zu konkreten vertrauensunterstützenden funktionalen Vorgaben zu gelangen, liegt in den folgenden Methodenschritten der Schwerpunkt mehr auf technischen Orientierungspunkten, wie Ansätzen des Requirements Engineering als auf Arbeiten zu Vertrauen.

Im Anforderungsmanagement wird zwischen funktionalen und nichtfunktionalen Anforderungen unterschieden. Die funktionalen Anforderungen definieren die einzelnen vom IT-System bereitzustellenden Funktionen und sind die Grundlage des weiteren Entwicklungsprozesses [37]. Nichtfunktionale Anforderungen sind nicht direkt mit Systemfunktionen verbunden. Sie spezifizieren oder beschränken Systemeigenschaften des IT-Systems wie Verfügbarkeit, Sicherheit oder Rechtmäßigkeit [67]. Bei den im vorangegangenen Schritt identifizierten Vertrauensdeterminanten handelt es sich um unterspezifizierte funktionale Anforderungen. Sie sind eine Teilmenge der nichtfunktionalen Anforderungen, die zur Verwendung in der weiteren Entwicklung durch eine entsprechende Detaillierung in funktionale Anforderungen überführt werden müssen [37]. Die für die konkrete Umsetzung eines IT-Systems benötigten funktionalen Anforderungen beschreiben die einzelnen Funktionen des zu entwickelnden IT-Systems und stellen die Grundlage in jedem Entwicklungsprozess dar. Zur Überführung nichtfunktionaler Anforderungen in funktionale Anforderungen gibt es mehrere Ansätze im Anforderungsmanagement [68], [69], [70], [71], [37].

Das hier genannte Beispiel folgt der von [37] vorgeschlagenen Detaillierung. Wichtig bei der Ableitung vertrauensbezogener funktionaler Anforderungen ist, dass die Situationsabhängigkeit von Vertrauen berücksichtigt wird. Das bedeutet, dass nicht nur die in Teilaktivität 3.2.4 abgeleitete Determinante mit einbezogen werden muss, sondern auch der Zeitpunkt im Verwendungsprozess, in dem die zugehörige Unsicherheit auftritt (Ergebnis der ersten Teilaktivität). Für die Determinante Verständlichkeit im Beispiel des Navigationsdienstes ergeben sich dabei folgende funktionale Anforderungen:

Der Nutzer sollte im Zeitpunkt der Präsentation der Route bei Bedarf Zusatzinformationen abrufen können, die darlegen, warum das System glaubt, dass dies aktuell die für ihn beste Route ist. Zu den Zusatzinformationen gehören Baustellen, Staus, Benzinverbrauch und voraussichtlich benötigte Zeit für die vorgeschlagene und andere mögliche Routen.

Ergebnis der vierten Teilaktivität ist eine Liste funktionaler Anforderungen an das zu entwickelnde soziotechnische ubiquitäre System. Diese sind durch die Detaillierung der identifizierten Vertrauensdeterminanten im Hinblick auf den Verwendungsprozess entstanden. Die Detaillierung führt zu mindestens einer funktionalen Anforderung je Vertrauensdeterminante. Da die Vertrauensdeterminanten so ermittelt wurden, dass sie direkt auf die beim Nutzer ermittelten Unsicherheiten wirken, verlangen jetzt die funktionalen Anforderungen Funktionen vom soziotechnischen ubiquitären System, die die Unsicherheiten reduzieren werden.

Beispiel Anforderungen der Vertrauenswürdigkeit für Meet-U

Die hier beschriebene Methode wurde bereits mehrfach angewendet. In [54] ist z. B. detailliert beschrieben, wie die Methode angewandt wurde, um vertrauensbezogene funktionale Anforderungen an den Dienst „Dinner Now“ abzuleiten. Zusätzlich wurde die Methode eingesetzt, um vertrauensbezogene funktionale Anforderungen an den VENUS-Prototyp „Meet-U“ abzuleiten. Tabelle 1

zeigt einen Auszug der identifizierten Vertrauensdimensionen, -determinanten, darauf basierende vertrauensbezogene Anforderung. Eine prototypische Darstellung, wie jedes einzelne Ergebnis der einzelnen Teilaktivitäten aussehen kann ist in [54] zu finden.

Vertrauensdimension	Vertrauensdeterminante	Anforderung(en)
Prozessnachvollzieh-barkeit	Verständlichkeit	Wenn vom Nutzer Daten verlangt werden, sollen Informationen einsehbar sein, die vermitteln, für welchen Zweck Meet-U diese Daten benötigt.
Prozessnachvollzieh-barkeit	Kontrolle	1. Der Nutzer sollte informiert werden, wenn Meet-U selbständig Anpassungen (z. B. Stummschalten) vornimmt. 2. Der Nutzer sollte die Möglichkeit haben, die selbständige Anpassung von Meet-U einfach zu widerrufen.
Zweckklarheit	Kommunikation des Einsatzzweckes	Jeder verfügbare Drittanbieterdienst muss eine vordefinierte Beschreibung anbieten, die den Nutzer über den genauen Sinn und Zweck sowie die Funktionsweise informiert.
Performanz	Informationsgenauigkeit	1. Der Nutzer sollte einsehen können, wie lange es her ist, dass jemand zu einer Veran-staltung zu- bzw. abgesagt hat. 2. Meet-U sollte einen Tag vor einem Event eine kurze Bitte an alle Teilnehmer schicken, Ihren aktuellen Status noch einmal zu kon-trollieren und ggf. anzupassen.

Tabelle 2: Zuordnung der Vertrauensdimensionen und Vertrauensdeterminanten zu den Unsicherheiten bei der Nutzung von Meet-U

3.3 VB: Normative Anforderungserhebung Gebrauchstauglichkeit

Besonderheiten der Anforderungserhebung Gebrauchstauglichkeit in der VENUS-Entwicklungsmethode:

- Ableitung von Anforderungen aus Normen der Gebrauchstauglichkeit

Produkt	Liste mit Nutzungsanforderungen
Verantwortliche Rolle	Usability Engineer
Mitwirkende Rollen	Anforderungsanalytiker Nutzer

Für ubiquitäre Systeme ergeben sich besondere Herausforderungen, die u.a. darin bestehen, dass die potentiellen Nutzer kein oder wenig Nutzungserfahrungen mit dieser Art von Systemen haben. Aus diesem Grund werden, anders als im Usability Engineering üblich, parallel zu nutzerbasierten Verfahren zur Anforderungsgewinnung, technische Anforderungen von Experten und ohne direkte Nutzerbeteiligung erhoben. Dieses Vorgehen ähnelt dem bei der Evaluation auf Gebrauchstauglichkeit üblichen heuristischen Evaluation nach Nielsen [72]: Experten arbeiten mit normativ festgelegten Vorgaben. Anders als bei der Evaluation werden im Rahmen der Anforderungserhebung für ubiquitäre Systeme normative Vorgaben herangezogen aus denen Anforderungen an die technische Gestaltung abgeleitet werden. Für diesen Zweck wurde die Methode zur Konkretisierung rechtlicher Anforderungen KORA (vgl. Abschnitt 3.1) aus dem rechtswissenschaftlichen Bereich auf das Usability Engineering übertragen [73].

3.3.1 Aktivität: Normative Anforderungserhebung durch Usability-Experten

Die Struktur der Methode KORA (vgl. Abschnitt 3.1) wurde auf die ergonomische Technikgestaltung übertragen. Auch im Bereich der Ergonomie existieren normative Vorgaben, die einen Soll-Zustand beschreiben, dessen Realisierung in einem technischen System möglichst nachvollziehbar zu gestalten ist. Bei KORA werden die rechtlichen Vorgaben (Stufe 1) aus Rechtsnormen abgeleitet. In der Ergonomie werden die Vorgaben aus Dokumenten der nationalen und internationalen Normung entnommen und abgeleitet. Es besteht ein umfangreiches Portfolio an Normen, die auf die ergonomische Gestaltung ausgerichtet sind. Die Normenfamilie ISO 9241 „Ergonomie der Mensch-System-Interaktion“ stellt in diesem Zusammenhang das wichtigste Bezugssystem dar und soll zukünftig in ihrer Bedeutung sukzessive weiter ausgebaut werden. Entsprechend zu KORA wird es darauf ankommen, auf der Grundlage der zuvor erstellten Anwendungsszenarien, des Geschäftsmodells und der Chancen- und Risikenanalyse eine Auswahl der relevanten Normen zu treffen und aus diesen ergonomische Vorgaben herzuleiten. Die Stufe 2, die Konkretisierung dieser Vorgaben zu Kriterien kann in den meisten Fällen aus den Normen übernommen werden. Die Normen beinhalten Konkretisierungen der übergeordneten Vorgaben, die Kriterien im Sinne von KORA entsprechen. Die Kriterien stellen noch keine Lösungsansätze dar, sondern beschreiben die Voraussetzungen, die von der später zu realisierenden Technik erfüllt werden sollen. Beispiele für solche Kriterien sind die Dialogprinzipien aus ISO 9241-110, die als konkrete Kriterien für die Vorgabe einer guten Dialoggestaltung dienen. Der nun folgende, dritte Schritt der Übersetzung der gefundenen Kriterien in technische Anforderungen stellt den Übergang zur Umsetzung der ergonomischen Normen in Gestaltung dar, die im folgenden Designprozess konkret umgesetzt wird. In den abgeleiteten technischen Anforderungen werden erstmals mögliche Lösungen adressiert, allerdings ohne sich dabei auf konkrete

Technikmerkmale festzulegen. Das beschriebene Vorgehen ist iterativ, d.h. die Stufen werden nicht immer in der gleichen Abfolge durchlaufen. Es kommt auch vor, dass ausgehend von einer technischen Anforderung auf Kriterien, Vorgaben und die entsprechenden Normtexte zugeordnet werden, oder dass sich bei der Konkretisierung neue Problemfelder erschließen, die ergänzt werden.

Beispiel Anforderung der Gebrauchstauglichkeit für Meet-U

Zum besseren Verständnis im Folgenden ein Beispiel aus dem ergonomischen Bereich für den beschriebenen dreistufigen Prozess entsprechend der Vorgehensweise von KORA. Die Vorgabe der Eignung von Multimediaschnittstellen für Wahrnehmung und Verständnis wird aus ISO 14915 abgeleitet: Die Anwendung muss für Wahrnehmung und Verständnis geeignet sein, also leicht erfasst und verstanden werden können. Aus dieser Vorgabe lässt sich folgendes Kriterien zur Vermeidung von Wahrnehmungsüberlastung aus der Norm ableiten: Der Nutzer sollte nicht mit zu vielen Informationen überlastet werden. Anschließend wird die abzuleitende technische Anforderung formuliert: Das System soll Informationen für den Nutzer zu einer Zeit immer nur über einen Wahrnehmungskanal und dezent anbieten.

3.4 VB: Normative Anforderungserhebung Softwarequalität

Besonderheiten der Anforderungserhebung Softwarequalität in der VENUS-Entwicklungsmethode:

- Ableitung von Anforderungen aus Normen der Softwarequalität
- Überführung nichtfunktionaler Anforderungen in funktionale Anforderungen

Produkt	Sammlung normativer Anforderungen aus Sicht der Informationstechnik
Verantwortliche Rolle	System Engineer
Mitwirkende Rollen	Anforderungsanalyst

Die Erhebung normativer Anforderungen aus Sicht der Informatik kann für den Bereich der Softwarequalität stattfinden. Der Prozess orientiert sich an der Methode zur Konkretisierung rechtlicher Anforderungen zu technischen Gestaltungsvorschlägen für Informations- und Kommunikationssysteme (KORA, vgl. Abschnitt 3.1). Hierbei werden vorhandene normative Vorgaben in normative Kriterien überführt, um anschließend daraus konkrete technische Anforderungen (funktionale Anforderungen) abzuleiten. Als Quellen kann die ISO 25010 (2011) dienen.

Zusätzlich ergeben sich Anforderungen des Kunden an die Lieferungen und Leistungen des Auftragnehmers. Im Gegensatz zu funktionalen Anforderungen, die beschreiben was ein System leisten soll (funktional), geben nichtfunktionale Anforderungen an, wie gut ein System etwas leisten soll (qualitativ). Technische nichtfunktionale Anforderungen können z. B. beschreiben, wie zuverlässig eine Applikation laufen soll, ob die Applikation von mehreren Nutzern gleichzeitig verwendbar sein soll und z. B. ob die Applikation nutzerspezifischen Daten verarbeiten soll oder nicht. Diese technischen nichtfunktionalen Anforderungen lassen sich nicht direkt vom Entwickler in der Implementierungsphase umsetzen, sondern müssen zuerst näher spezifiziert werden. Auch hier kann die Schrittweise Konkretisierung der Anforderung durchgeführt werden.

3.5 VB: Anforderungsvereinbarung

Besonderheiten der Anforderungsvereinbarung in der VENUS-Entwicklungsmethode:

- Interdisziplinäre Vereinbarung

Produkt	Liste mit eindeutig vereinbarten Anforderungen
Verantwortliche Rolle	Anforderungsanalyst
Mitwirkende Rollen	Diverse

Die Anforderungsvereinbarung dient dazu, Konflikte innerhalb der Anforderungen zu identifizieren und Lösungen zur Beseitigung der Konflikte herauszuarbeiten. Nach [37, p. 396] besteht innerhalb des Anforderungsmanagements ein Konflikt, „wenn die Bedürfnisse und Wünsche verschiedener Stakeholder an das geplante System sich widersprechen oder nicht alle Bedürfnisse und Wünsche berücksichtigt werden“. Hierfür bedarf es nach [74] zunächst das Verständnis über die Anforderungen zu klären und daraufhin diese zu priorisieren sowie Bedenken zu ermitteln. Um dann im Anschluss Lösungsvorschläge zu erarbeiten und diese zu evaluieren.

Hierzu ist in diesem Vorgehensbaustein die Anwendung der EasyWinWin Methode zielführend [74]. Das EasyWinWin Verfahren ist eine Groupware-gestützte, kollaborative Methode, zur Verhandlung von Anforderungen. Ziel ist, die Anforderungen für das Projekt eindeutig und konkret festzulegen. Dies geschieht in einem Workshop mit den beteiligten Stakeholdern und wird in den folgenden Teilaktivitäten vorgestellt. Als Ausgangslage und Input für diesen Vorgehensbaustein dient eine Liste mit den vorhergehenden Vorgehensbausteinen erhobenen Anforderungen sowie die Anforderungen der Stakeholder. Für die spätere Nachvollziehbarkeit sollte die jeweilige Quelle der Anforderungen ersichtlich bleiben.

3.5.1 Aktivität: Verständnis klären

In dieser Teilaktivität wird das Verständnis der Anforderungen geklärt. Der Anforderungsanalyst dient dabei als Moderator und führt durch den Prozess. In der Anforderungserhebung der verschiedenen Disziplinen können irrelevante und redundante Ideen entstehen, aus denen die erfolgskritischen Anforderungen abgeleitet werden müssen. Hierfür werden die Ideen von der Gruppe in vordefinierte Kategorien eingeordnet. So wird die Komplexität der Anforderungserhebung erheblich reduziert und die Ideen klarer formuliert. In der Regel entstehen innerhalb einer Projektarbeit eine gewisse Insidersprache und Projektbezogene „Fachbegriffe“. Das kann zwar die Kommunikation innerhalb der Anwender deutlich vereinfachen, aber bei der Kommunikation mit Personen, die diese Insidersprache nicht beherrschen, zu Kommunikationsproblemen führen. Um eine gemeinsame sprachliche Basis zu finden, wird dieser Schritt durchgeführt. Hierfür wird eine gemeinsame Liste erstellt, welche den Projektjargon enthält. Dazu werden Begriffe samt Definition auf Vorschlag der Stakeholder in die Liste aufgenommen. Diese Begriffe und deren Definition müssen von jedem Teammitglied bestätigt werden. Erst nach Schaffung dieser gemeinsamen Basis kann eine Beurteilung stattfinden, ob die zuvor in den verschiedenen Disziplinen ermittelten Anforderungen redundanzfrei sind, also ob sich der Inhalt mit den eigenen Anforderungen überschneidet.

Es entsteht eine neue bereinigte Liste mit Anforderungen. Die Stakeholder nennen aus der ursprünglichen Liste Anforderungen, die sie für wichtig halten und die noch nicht in der bereinigten vorhan-

den sind. Wichtig ist dabei, dass die beteiligten Stakeholder die Liste des Projektjargons berücksichtigen. Nach jedem Vorschlag eine Anforderung in die neue Liste aufzunehmen findet zunächst eine Überprüfung des Verständnisses der Stakeholder statt. Erst nach Klärung des Verständnisses und Zustimmung der anderen Stakeholder wird die Anforderung in die zweite Liste aufgenommen. Sollte eine Anforderung nicht verstanden werden, wird sie umformuliert, bis alle Stakeholder sie verstanden haben. Die Teilaktivität endet, wenn die Stakeholder keine Anforderungen mehr zum Übernehmen vorschlagen.

3.5.2 Aktivität: Anforderungen priorisieren

In der zweiten Teilaktivität besteht das Ziel, die Anforderungen nach zwei verschiedenen Kriterien zu bewerten. Zum einen werden die Anforderungen nach sogenannter „Business Importance“ [74] bewertet. Das bedeutet, dass eine Bewertung stattfindet, zu welchem Grad der Erfolg des Projektes von der Umsetzung der Anforderung abhängt. Es wird demnach die Wichtigkeit der Umsetzung bewertet. Der zweite Gesichtspunkt der Bewertung beschäftigt sich mit der „Ease of Realization“ [74], also der Einfachheit der Umsetzung. Die Teilnehmer bewerten eine Anforderung, zu welchem Grad diese technologisch, sozial, politisch und ökonomisch umsetzbar ist.

Die Bewertung geschieht nach dem Prinzip: „If you don’t know, don’t vote“ [74]. Das bedeutet, dass die Teilnehmer nur bewerten sollten, wenn es um deren Fachbereich und Interesse geht. So kann beispielsweise das Prinzip einer Likert-Skala [75], also personenorientierte, rangskalierte Bewertung angewandt werden. Die Anforderungen mit einer hohen Standardabweichung werden auf Verständnisunterschiede besprochen und, wenn nötig, angepasst. Dazu legen die Personen, die für die Anforderung gegensätzliche Bewertungen abgegeben haben, ihr Verständnis über die Anforderung und die Begründung für ihre Einschätzung dar. Die Teilaktivität Anforderungen priorisieren ist nicht dafür vorhanden, schlecht bewertete Anforderungen auszuschalten, sondern um Meinungsverschiedenheiten zu den einzelnen Anforderungen aufzuzeigen. Hierfür werden die Ergebnisse graphisch dargestellt. Aufkommende Verständnisprobleme müssen erneut diskutiert werden und gegebenenfalls alternative Formulierungen gefunden werden. Für die Erörterung der Gründe von Abweichungen in den Meinungen über die Anforderungen wird eine strukturierte Diskussion durchgeführt.

3.5.3 Aktivität: Anforderungen vereinbaren

Die Anforderungen können untereinander im Konflikt stehen, welche in diesem Schritt erörtert werden. Hierfür wird gemeinsam die Liste aller Anforderungen dreimal durchgangen. In den Schritten werden strukturiert Bedenken, Lösungsvorschläge und Übereinstimmungen zu den Anforderungen gesammelt.

- Bedenken ermitteln
  Im ersten Schritt liest jeder Teilnehmer jede Anforderung. Falls eine Anforderung bei einem Stakeholder ein Bedenken darstellt, schreibt dieser eine Notiz an die Anforderungen.
- Lösungsvorschläge erarbeiten
  Im zweiten Schritt lesen alle Teilnehmer die Bedenken der Stakeholder. Falls einem eine Lösung für das Problem in den Sinn kommt, schreibt dieser diese auf. Jeder Teilnehmer geht die Liste mit den Anforderungen und Bedenken durch und notiert Lösungsvorschläge für die Bedenken. Anforderungen ohne Bedenken werden in eine dritte Liste „abgestimmte Anforderungen“ verschoben.

- Lösungsvorschläge evaluieren
  In der Teilaktivität Lösungsvorschläge evaluieren wird über die Anforderungen und Bedenken verhandelt. Die Anforderungen mit Bedenken werden in der Gruppe besprochen. Dabei wird vor allem auf die Lösungsvorschläge eingegangen, die im vorherigen Schritt gesammelt wurden. Kann sich auf einen Lösungsvorschlag geeinigt werden, wird dieser in die finale Liste übernommen. Kann sich auf keinen Lösungsvorschlag geeinigt werden, sind verschieden Eskalationsstufen denkbar. Als erstes kann versucht werden, ein Kompromiss zu finden. Ist auch dies nicht möglich, muss eine Entscheidung durch eine übergeordnete Stelle ersucht werden.

Beispiel Anforderungsvereinbarung Meet-U

Die Anforderungsvereinbarung für Meet-U wird in im Folgenden verdeutlicht. Abbildung 6 zeigt das Ergebnis der Priorisierung mit Hilfe der Groupware ThinkTank. Dabei werden Anforderungen mit hoher Standardabweichung rot markiert. Eine hohe Standardabweichung deutet darauf hin, dass sich die Bewertungen der Stakeholder dieser Anforderung unterscheiden. Bei diesen Anforderungen sollte an dieser Stelle sichergestellt werden, dass das gleiche Grundverständnis über die Anforderung herrscht und, falls sich dabei Änderungen im Verständnis oder an der Anforderung ergeben, eine neue Bewertung vorgenommen werden. Bedenken werden im darauffolgenden Schritt geäußert. Abbildung 7 zeigt das Ergebnis der Anforderungsvereinbarung mit den dokumentierten bedenken und Lösungsvorschlägen.

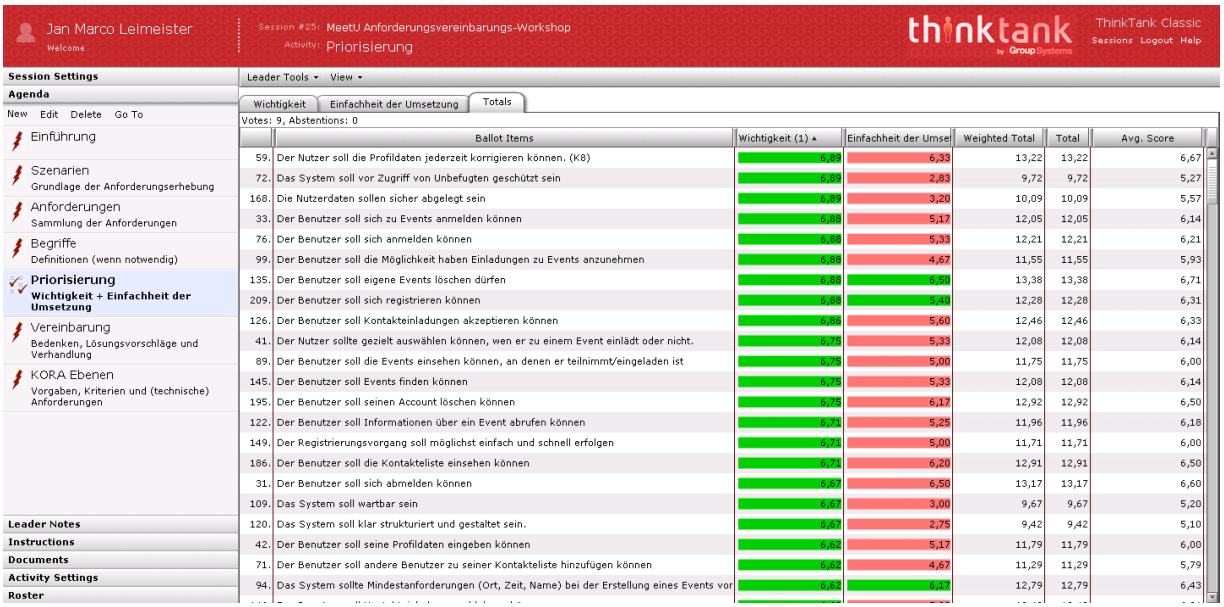


Abbildung 6: Ergebnis der Priorisierung der Anforderungen (Ausschnitt)

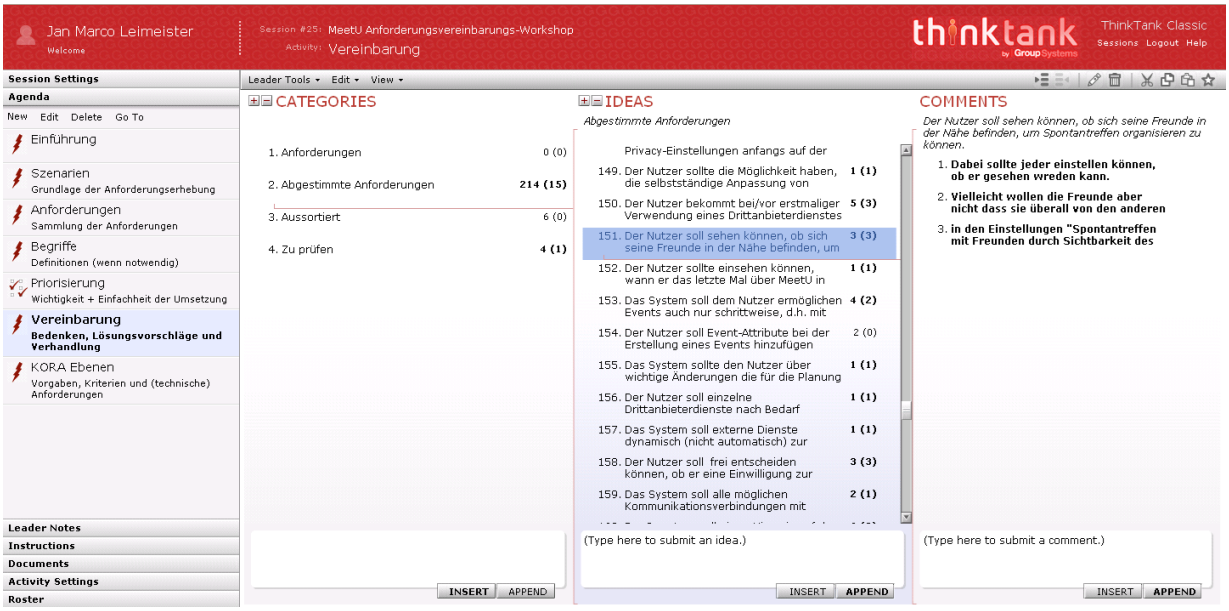


Abbildung 7: Ergebnis der Anforderungsvereinbarung mit den dokumentierten Bedenken und Lösungsvorschlägen (Ausschnitt)

3.6 Werkzeugunterstützung: Anforderungsmuster

Besonderheiten der Anforderungserhebung in der VENUS-Entwicklungsmethode:

- Wiederverwendbare Anforderungsmuster dokumentieren häufig auftretende soziotechnische Anforderungen

Produkt	Funktionale Anforderungen
Verantwortliche Rolle	Anforderungsanalyst
Mitwirkende Rollen	Trust Engineer Usability Engineer Jurist System Engineer

Um die Anforderungserhebung der beteiligten Disziplinen zu unterstützen, können Anforderungsmuster eingesetzt werden. Anforderungsmuster sind eine Art der strukturierten Wiederverwendung von Anforderungen Sie können Anforderungsanalysten dabei helfen, Anforderungen zu identifizieren und zu dokumentieren [76]. Sie helfen vor allem dabei, regelmäßig benötigte Anforderungen wiederzuverwenden [77]. Anforderungsmuster unterscheiden sich in die Typen *Process* und *Product*. Erstere beschreiben zum Beispiel notwendige Schritte zur Durchführung einer Aktivität in der Anforderungserhebung. So kann diese Anleitung zur Methodik bereits als Muster gesehen werden. Letztere hingegen enthalten Artefakte zum Einsatz in den verschiedenen Aktivitäten [78]. Zu Bereitstellung von wiederverwendbaren Anforderungen werden Muster vom Typ *Product* benötigt. Ein Beispiel für ein Anforderungsmuster zur Vertrauenswürdigkeit zeigt Tabelle 3.

Anforderungsmuster enthalten neben Metadaten ganz konkrete Anforderungen, die in die Anforderungsdokumentation übernommen werden kann. Die Anforderungen verfolgen dabei ein im Anforderungsmuster angegebenes Ziel. Anhand des Ziels kann der Anforderungsanalyst passende Anforderungsmuster auswählen. Für die entsprechende Anforderung kann er dann entweder direkt die standardisierte Anforderung, eine angegebene Erweiterung oder eine Modifikation der Anforderung nutzen. Durch die angegebenen Beziehungen ergeben sich weitere Anforderungsmuster, die der Anforderungsanalyst betrachten sollte. Auch auf Konflikte wird hingewiesen {für eine ausführliche Vorgehensweise siehe [79].

Eine auf Anforderungsmuster basierende Anforderungserhebung kann den Aufwand für die eingesetzten Experten verringern. Vorteil ist nicht nur die ersparte Zeit, sondern auch die einheitliche Formulierung und geprüfte Qualität der Anforderungen in den Anforderungsmustern [80]. Dabei sollten Anforderungsmuster das Wissen aus vorherigen Projekten bereitstellen und ein entsprechender Anforderungsmusterkatalog kontinuierlich aktualisiert werden [79]. Bestehende Kataloge für die soziotechnischen Anforderungen existieren für die Rechtsverträglichkeit [81], die Vertraulichkeit [82], [83] und die Qualitätseigenschaften von Software [84].

Anforderungsmuster: Informationen zu Funktionen			
Metadaten	Ziel	Die Nutzer wissen, was die Applikation bei einer bestimmten Eingabe macht, warum und wie sie das macht.	
	Grundlage	*Predictability* [64]	
	Abhängigkeiten	-	
	Verknüpfungen	V-S-02: Signalisieren des Funktionsstatus V-S-04: Erklärung von Vorgängen	
	Konflikte	-	
Vorlage Funktion	Standardisierte Anforderung	Die Applikation soll die Nutzer bei Bedarf vor der Ausführung einer Funktion über diese Funktion informieren.	
	Erweiterung	Die Applikation soll die Nutzer bei Bedarf vor der <Funktion> darüber informieren, <Information>.	
		Parameter	Werte
		<Funktion>	Ausführung von [Funktion der Applikation] [Dateneingabe]
		<Information>	was <Funktion> macht warum Applikation <Funktion> bereitstellt wie <Funktion> funktioniert
	Hinweise	Die Information beginnt mit einer prägnanten Beschriftung von Schaltflächen und sollte darüber hinaus in einem Info-Feld bereitgestellt werden.	

Tabelle 3: Anforderungsmuster [83]

4 Konzeptdesign (Phase 3)

Besonderheiten der Phase Konzeptdesign in der VENUS-Entwicklungsmethode:

- Berücksichtigung der speziellen Benutzungsschnittstellen ubiquitärer Systeme
- Berücksichtigung der Auswirkungen automatischer Adaptionen der Applikation
- Werkzeugunterstützung durch interdisziplinäre Designmuster zur Umsetzung soziotechnischer Anforderungen

In der dritten Phase der VENUS-Entwicklungsmethode wird das Konzeptdesign des ubiquitären Systems erarbeitet. Unter dem Begriff Konzeptdesign wird der konzeptionelle und gestalterische Entwurf der Benutzungsschnittstelle des Systems verstanden. Eine Benutzungsschnittelle ist nach DIN EN ISO 9241-210 definiert als: „Alle Bestandteile eines interaktiven Systems (Software oder Hardware), die Informationen und Steuerelemente zur Verfügung stellen, die für den Nutzer notwendig sind, um eine bestimmte Arbeitsaufgabe mit dem interaktiven System zu erledigen.“ [85] Ubiquitäre Systeme sind adaptiv, d.h. sie passen sich an den jeweiligen Nutzer, die vorherrschende Situationen oder den Nutzungskontext an. Diese Anpassungen können Änderungen in der Benutzungsschnittstelle mit sich ziehen. Sie müssen im Konzeptdesign ebenso erarbeitet werden. Außerdem kann es für ubiquitäre Systeme erforderlich sein, das das Konzeptdesign mehrere Benutzungsschnittstellen umfasst.

Die Entwicklung der Benutzungsschnittstelle erfolgt in verschiedenen Aktivitäten, die aufeinander aufbauende Artefakte als Ergebnis haben. Diese Artefakte definieren die verschiedenen Ebenen der Benutzungsschnittstelle. Eine Benutzungsschnittstelle kann in verschiedene aufeinander aufbauende Ebenen aufgegliedert werden. Verschiedene Autoren tun dies in unterschiedlichen Granularitäten. Garret [86] listet z. B. fünf Ebenen in seinem Modell zur User Experience auf. Baxley [87] dagegen beschreibt ein universelles Modell einer Benutzungsschnittstelle und gibt neun Ebenen mit drei Oberebenen an. Koch et al. [88] geben in ihrem Schnittstellenmodell sieben Ebenen an. Abbildung 8 gibt einen Überblick über die Ebenen einer Benutzungsschnittstelle, die im Rahmen der in diesem Kapitel beschriebenen Designaktivitäten erarbeitet werden. In dieser Phase werden die Ebenen von unten nach oben, d.h. vom Abstrakten zum Konkreten aufeinander aufbauend konstruiert. Das Konzeptdesign basiert grundsätzlich auf den Anforderungen, die in der vorangegangenen Phase definiert wurden. Anforderungen bilden also das Fundament für Funktionsstruktur und Informationsarchitektur, Funktionslayout und die detaillierte Ausarbeitung von visueller Gestaltung usw.

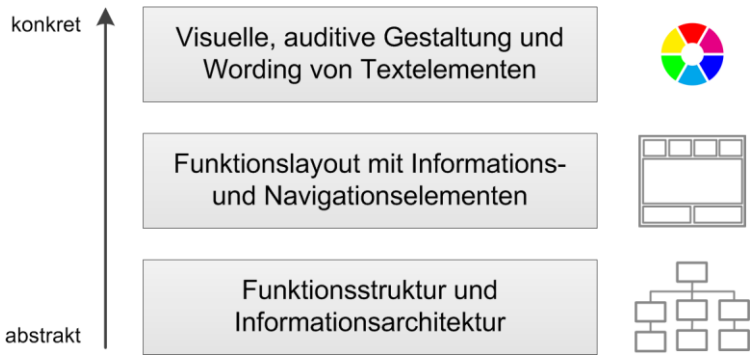


Abbildung 8: Überblick über Ebenen der Benutzungsschnittstelle, die in den Designaktivitäten erarbeiten werden

Das Konzeptdesign wird schrittweise erarbeitet. Für eine effizientere Erarbeitung des Konzeptdesign können Entwurfsmuster (z. B. in [89]) in den Design Aktivitäten als Werkzeugunterstützung dienen (siehe Kapitel 4.2). Entwurfsmuster fassen die bereits erfolgten Abstimmungsprozesse zusammen, weshalb diese dann in geringerem Ausmaß erforderlich sind.

4.1 VB: Design

Besonderheiten der Designaktivitäten in der VENUS-Entwicklungsmethode:

- Berücksichtigung der speziellen Benutzungsschnittstellen ubiquitärer Systeme
- Berücksichtigung der Auswirkungen automatischer Adaptionen der Applikation

Produkt	Konzept-Design der Benutzungsschnittstelle
Verantwortliche Rolle	Usability Engineer
Mitwirkende Rollen	Trust Engineer Usability Engineer Jurist Nutzer

In den Designaktivitäten wird das Konzeptdesign einer ubiquitären Benutzungsschnittstelle schrittweise durch verschiedene Teilaktivitäten erarbeitet. Für die Entwicklung eines ubiquitären Systems können auf Basis der Anforderungen verschiedene separate oder funktional zusammenhängende Benutzungsschnittstellen zu erarbeiten sein. In jeder Teilaktivität wird eine der aufeinander aufbauenden Ebenen einer Benutzungsschnittstelle erarbeitet. Dabei sind teilweise mehrere Teilaktivitäten für eine Ebene erforderlich sein. Einen Überblick über die in den Teilaktivitäten zu erarbeitenden Artefakte und ihr Zusammenhang im Designprozess gibt die Abbildung 9.

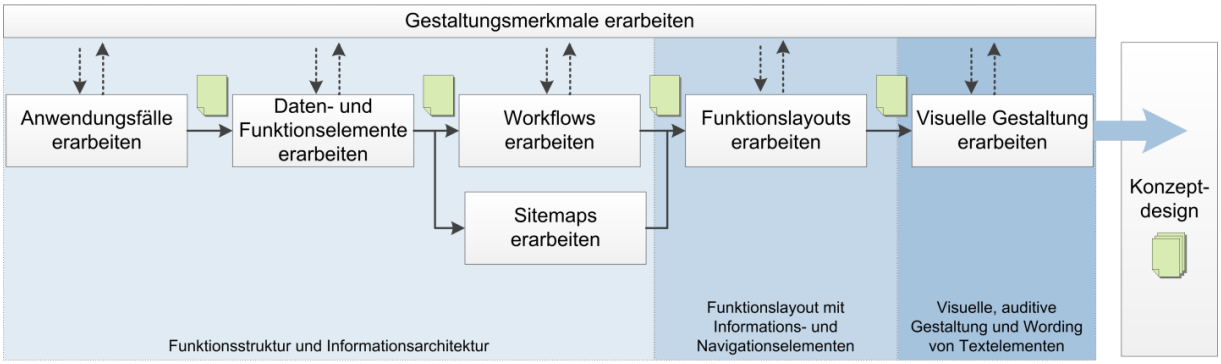


Abbildung 9: Überblick über den Zusammenhang Artefakt-erzeugenden Designaktivitäten

In jeder Teilaktivität erfolgt bedarfsbedingt eine Rückkopplung mit dem Expertenteam, um die Einarbeitung von normativen Vorgaben und Design-Prinzipien hinsichtlich Rechtsverträglichkeit, Vertrauensunterstützung und Gebrauchstauglichkeit effektiv zu gewährleisten. Diese Erarbeitung von Gestaltungsmerkmalen erfolgt übergreifend in der Aktivität.

4.1.1 Aktivität: Gestaltungsmerkmale erarbeiten

Aus dem Anforderungserhebungskatalog ergeben sich technische Anforderungen, die noch relativ abstrakt sind. Um ein schlüssiges sozialverträgliches Konzeptdesign ausarbeiten zu können, müssen konkrete Gestaltungsmerkmale einzelner Systembausteine erarbeitet werden. Diese können dann in ein konkretes Design der Benutzungsschnittstelle überführt werden. Die Ausarbeitung dieser Gestaltungsmerkmale erfolgt nicht als einzelner Arbeitsschritt, sondern ist Teil der Erarbeitung der einzelnen Artefakte des Konzeptdesigns und fließt in die anderen Teilaktivitäten der hier beschriebenen Aktivität zur Erarbeitung des Konzeptdesigns ein.

Diese Aktivität entspricht dem letzten Schritt der Methode KORA (siehe dazu etwa [49, p. 46 f.], [50]). Besteht im Hinblick auf einzelne Anforderungen oder Gruppen von Anforderungen die Möglichkeit, diese zu alternativen Gestaltungsvorschlägen zu konkretisieren, so muss für das Konzeptdesign letztlich der Gestaltungsvorschlag ausgewählt werden, durch den, unter Berücksichtigung des Gesamtsystems, die Anforderungen bestmöglich umgesetzt werden. Ziel ist also die Erreichung einer möglichst sozialverträglichen Lösung bei gleichzeitiger technischer Realisierbarkeit. Der ausgewählte Gestaltungsvorschlag wird zum Gestaltungsmerkmal des zu entwickelnden Systems, das zur Erarbeitung des konkreten Designs in die Teilaktivität getragen wird.

Weil die Anforderungen die Vorgaben und Wünsche aller Beteiligten und damit auch der beteiligten Experten umsetzen, können diese unmittelbar zu einem sozialverträglichen Konzeptdesign verdichtet werden. Die Erarbeitung der Gestaltungsvorschläge und die Auswahl einzelner Gestaltungsmerkmale spielt im Rahmen der einzelnen Aktivitäten des Konzeptdesigns eine Rolle. Eine getrennte Erarbeitung von Gestaltungsvorschlägen durch einzelne Experten ist nicht notwendig und häufig auch kaum noch möglich, da einzelne oder mehrere Anforderungen, aus denen sich schließlich ein Gestaltungsmerkmal ergibt, nicht zwingend von nur einem der Beteiligten stammen. Die Gestaltungsmerkmale sind deshalb durch die für den Usability Engineer zu erarbeiten. Da für die Anforderungen eine einheitliche Sprache gefunden und spätestens bei der Anforderungsvereinbarung (vgl. Abschnitt 3.5) Verständnisprobleme geklärt wurden, ist dieser nun in der Lage, die Anforderungen zu Gestaltungsmerkmalen zu konkretisieren und diese in das Konzeptdesign einfließen zu lassen.

Ist es dem Usability Engineer nicht möglich, mit Hilfe der Anforderungen konkrete Gestaltungsmerkmale zu erarbeiten, können einzelne Mitwirkende hinzugezogen werden, um die Anforderungen gemeinsam zu Gestaltungsmerkmalen zu konkretisieren. Dabei ist vor allem an diejenigen Mitwirkenden zu denken, die die nun umzusetzenden Anforderungen gefordert haben. Nicht zuletzt aus diesem Grund ist es wichtig, dem Anforderungskatalog entnehmen zu können, von welchen Beteiligten eine Anforderung erhoben wurde (vgl. Abschnitt 3.5). Der Usability Engineer kann die Mitwirkenden außerdem zur Begutachtung einzelner Gestaltungsmerkmale oder –wie dies im Vorgehensbaustein Expertenvalidierung (vgl. Abschnitt 5.4) vorgesehen ist – des gesamten Konzeptdesigns hinzuziehen. Auch hierdurch kann sichergestellt werden, dass die erhobenen Anforderungen im Sinne einer sozialverträglichen Gestaltung möglichst optimal zu Gestaltungsmerkmalen konkretisiert werden und in dieser Form in das Konzeptdesign einfließen.

Beispiel für abgestimmtes Gestaltungsmerkmal

Bei der Gestaltung von Meet-U ist ein Impressum einzubinden. Im Rahmen der Teilaktivitäten Workflows und Sitemaps erarbeiten sind die Zugriffsmöglichkeiten auf diese Informationen zu definieren. Dabei sind unterschiedliche, sich zum Teil entgegenstehende Anforderungen zu beachten. So müssen die Informationen einerseits unmittelbar erreichbar sein, andererseits dürfen sie den Nutzer aber auch nicht stören. Für die Umsetzung ist auch zu berücksichtigen, auf welchem Ausgabemedium (Smartphone, Tablet, Laptop) eine Anwendung verwendet werden soll. Die Anwendung Meet-U soll auf Smartphones eingesetzt werden.

Relevante technische Anforderungen sind:

- Der Nutzer soll sich jederzeit informieren können, von wem ihm der Dienst angeboten wird (Umsetzung von Trust- und Recht-Vorgaben).
- Der Dienst muss ein Impressum enthalten, dessen Informationen für den Nutzer leicht erkennbar, unmittelbar erreichbar und ständig verfügbar sind (Umsetzung von Recht-Vorgaben).
- Bereitgestellte Informationen, die nicht aufgabenspezifisch sind, sollen die durch den Dienst unterstützte Bewältigung bestimmter Aufgaben nicht erschweren oder stören (Umsetzung von Usability-Vorgaben).

Alternative technische Gestaltungsvorschläge sind:

- Auf jeder Seite des Dienstes wird der Zugriff zum Impressum durch ein Icon am oberen Rand ermöglicht.
- Auf jeder Seite des Dienstes wird der Zugriff zum Impressum durch ein Icon am unteren Rand ermöglicht.
- Im Optionsmenü, das bei jeder Seite des Dienstes geöffnet werden kann, wird ein Icon mit Link zum Impressum eingebunden.
- Auf der Hauptseite des Dienstes wird am oberen Rand ein Icon mit Link zum Impressum eingebunden.
- Auf der Hauptseite des Dienstes wird am unteren Rand ein Icon mit Link zum Impressum eingebunden.
- Im Optionsmenü findet sich ein Icon mit Link zu rechtlichen Informationen, von dort wird mit einem weiteren Icon auf die Impressumsinformationen weiter verlinkt.

Ausgewähltes Gestaltungsmerkmal

- Im Optionsmenü, das bei jeder Seite des Dienstes geöffnet werden kann, wird ein Icon mit Link zum Impressum eingebunden.

Auch wenn bei der isolierten Betrachtung einzelner Anforderungen diese möglicherweise durch andere Gestaltungsvorschläge besser verwirklicht werden könnten, so führt die Gesamtbetrachtung dazu, dass die Umsetzung durch dieses Gestaltungsmerkmal die Anforderungen in ihrer Gesamtheit bestmöglich verwirklichen. Einerseits wird der Nutzer nicht durch störende Informationen oder Icons gestört, andererseits kann er das Impressum aber stets recht einfach und schnell aufrufen. Zu berücksichtigen war hier insbesondere auch, dass der Dienst auf einem Smartphone angeboten werden soll, das nur eine kleine Displaygröße zur Darstellung der aufgabenbezogenen Informationen zur Verfügung stellt. Das Display sollte deshalb nicht für aufgabenferne Informationen vereinnahmt werden.

4.1.2 Aktivität: Anwendungsfälle erarbeiten

Anwendungsfälle konkretisieren die gesammelten Anforderungen und Anwendungsszenarien (vgl. Abschnitt 2.5 zu genaueren Abläufen in der Anwendung). Dies hilft vor allem dabei den Funktionsumfang der Anwendung zu bestimmen und dient als Vorarbeit für die weiteren Designschritte. Anwendungsfälle bleiben dabei in der Regel auf abstraktem Niveau und beschreiben keine Details technischer Lösungen.

Erster Schritt bei der Erarbeitung der Anwendungsfälle ist deren Identifikation aus dem Anforderungsdokument. Insbesondere können hier neben den Anforderungen auch die Anwendungsszenarios (vgl. Abschnitt 2.5) relevant sein. Beide Quellen implizieren Nutzeraufgaben in der Anwendung, die in einem Anwendungsfall-Diagramm gesammelt werden, wie in Abbildung 10 zu sehen. Diese sind nach dem Nutzerziel benannt, z. B. „Anmeldung“. Die Granularität und der Detailgrad des Anwendungsfalls können dabei sehr stark variieren. Schließlich werden für jeden Anwendungsfall die beteiligten Akteure identifiziert. Diese werden im Diagramm mit dem Anwendungsfall verknüpft.

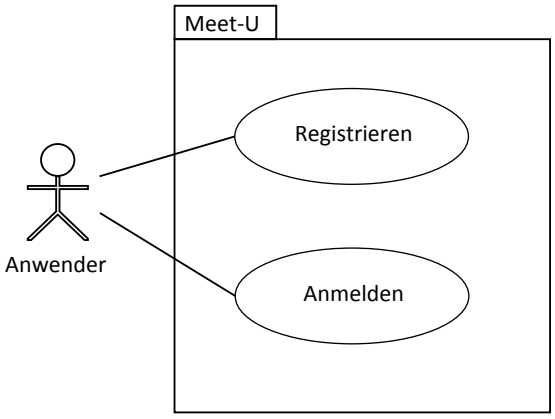


Abbildung 10: Anwendungsfall-Diagramm

Adaptionen sind Anpassungen des ubiquitären Systems an den Nutzer, Kontext o.ä. Adaptionen sollten bereits in den Anwendungsfällen berücksichtigt werden. Der Anwendungsfall sollte deutlich machen, aufgrund welcher Bedingung die Adaptation durchgeführt wird.

Im zweiten Schritt werden die gesammelten Anwendungsfälle der Anwendungsfall-Diagramme in textueller Form beschrieben. Dies beinhaltet insbesondere die Identifikation der genauen Vor- und Nachbedingungen, Ablaufbeschreibung und Alternativen.

Die folgende Schablone kann als Vorlage für diese Ausarbeitungen dienen:

Anwendungsfall-Name: Name des Anwendungsfalles.

Zusammenfassung: Gibt eine grobe Übersicht, was im Anwendungsfall beschrieben wird.

Abhängigkeiten: Spezifizieren die Beziehungen zu anderen Anwendungsfällen. Ein Anwendungsfall kann einen anderen beinhalten (include-Relation) oder Alternativen besitzen (extends-Relation).
Anmerkung: Extends werden verwendet, um die Komplexität zu reduzieren – primitive Alternativen können direkt beschrieben werden.

Akteure: Die beteiligten Akteure werden aufgelistet. Beispiele für Akteure sind: Nutzer, Administrator, Drittdienstanbieter, etc. Diese gehen z.T. bereits aus dem Anwendungsfalldiagram hervor.

Vorbedingungen: Es werden die Bedingungen beschrieben, die erfüllt sein müssen, bevor der Anwendungsfall ausgeführt wird.

Beschreibung: Diese enthält eine Erklärung der Aufgaben und deren Reihenfolge, welche der Akteur durchführt. Es wird der Regelfall beschrieben, wie der Akteur mit dem System interagiert.

Alternativen: Es werden die Alternativen beschrieben, welche auftreten können, jedoch nicht den Regelfall darstellen z. B. dass eine Falscheingabe des Nutzers das System wieder in die Ausgangssituation überführt.

Nachbedingungen: Die Bedingungen, die nach dem Durchführen des Anwendungsfalls erfüllt sind.

Offene Fragen: Wie in [90, p. 125] dargestellt können hier Fragen dokumentiert werden, die zur Diskussion mit Nutzern gedacht sind.

Zur Verdeutlichung können den Anwendungsfall-Beschreibungen auch Diagramme beigefügt werden, wie beispielsweise Sequenzdiagramme, um komplexere Anwendungsfälle zu verdeutlichen.

Dieses Vorgehen entspricht der Standardmodellierung der Anwendungsfälle, siehe [90, p. 119 ff.]. Die Anwendungsfälle stellen die Basis für ein einheitliches Verständnis der Disziplinen bezüglich der System-Funktionalitäten dar.

Des Weiteren wird das Anwendungsfallmodel von den Experten Trust Engineer, Usability Engineer und dem Jurist hinsichtlich Sozialverträglichkeit geprüft. Zwecks Unterstützung der Sozialverträglichkeit werden die Anwendungsfälle ergänzt oder modifiziert, um das Vertrauen in die Anwendung zu erhöhen und die Rechtskompatibilität zu gewährleisten. Nach Durchführung dieser Aktivität kann so sichergestellt werden, dass die beschriebene Funktionalität alle sozialverträglichen Aspekte berücksichtigt. Bei diesem Prozess kann zudem die Vollständigkeit der Anwendungsfälle geprüft und die Aktivität gegebenenfalls erneut durchgeführt werden.

Beispiel Anwendungsfall für Meet-U

Das folgende Beispiel zeigt den Anwendungsfall „Registrieren“:

Anwendungsfall Name: Registrieren

Zusammenfassung: Der Nutzer soll sich für Meet-U registrieren können, falls er noch kein Nutzerkonto besitzt. Der Nutzer wird dazu aufgefordert seine Daten (Name, Nutzername sowie Passwort und Email-Adresse) und optional seine Präferenzen einzugeben.

Akteure: Meet-U-Nutzer

Vorbedingungen: Meet-U ist gestartet, Nutzer ist nicht angemeldet.

Beschreibung: Der Nutzer ruft den Registrierungsdialog auf.
Der Nutzer wird darauf aufmerksam gemacht, in welcher Form er für den Dienst bezahlt (einmalige Zahlung von 3,99 Euro oder kostenlose Variante mit Schaltung personenbezogener Werbung und Verkauf von Daten). Der Nutzer muss die AGB von Meet-U bestätigen/einsehen können.
Die bestätigte AGB wird auf dem Smartphone gespeichert.
Der Nutzer gibt Nutzername, E-Mail Adresse und Passwort ein.
Soweit er sich für die „kostenlose“ Version entscheidet, wird der Nutzer aufgefordert, in die Verwendung der Daten für die Zwecke der Werbung und für den Datenverkauf einzuwilligen.

Alternativen: Der Nutzer möchte noch keine personenbezogenen Daten (außer Nutzername, Passwort, E-Mail) angeben. Dieser Schritt wird übersprungen.

Nachbedingungen: Nutzer hat einen Account erstellt und ist angemeldet.

4.1.3 Aktivität: Daten- und Funktionselemente erarbeiten

Daten- und Funktionselemente sind die für den Nutzer sichtbaren Hauptbausteine der konkreten Benutzungsschnittstelle. Datenelemente bezeichnen alle Elemente, mit denen der Nutzer während der Verwendung der Benutzungsschnittstelle interagiert. Funktionselemente bezeichnen alle Operationen, die der Nutzer während der Verwendung des Systems an den Datenelementen vornehmen kann. Daten- und Funktionselemente sind sichtbare Verkörperungen der funktionalen Anforderungen in der Benutzungsschnittstelle [91], die in der Bedarfsanalyse identifiziert wurden. Im Gegensatz dazu beschreiben Anwendungsfälle die Interaktion des Nutzers als Handlungsablauf aus der Sicht des Nutzers.

Durch die Definition der Daten- und Funktionselemente werden alle Grundbausteine der Benutzungsschnittstelle zusammengetragen. Die Menge an Daten- und Funktionselementen fasst alle Handlungsobjekte und Handlungsmöglichkeiten, die dem Nutzer bei der Nutzung des Systems zur Verfügung stehen zusammen. Sie unterstützt im folgenden Vorgehen im Konzeptdesign dabei, keine wichtigen Bausteine bei der Entwicklung der Benutzungsschnittstelle zu vergessen.

Datenelemente sind die fundamentalen Bausteine interaktiver Systeme, da der Nutzer mit ihnen interagiert. Das sind beispielsweise Fotos, Nachrichten, Kontakte usw. Datenelemente können andere Datenelemente enthalten, beispielsweise befinden sich Kontakte in einem Adressbuch. Die Beziehungen zwischen Datenelementen können auch inhaltlicher Art sein. Ein Datenelement besitzt zudem Eigenschaften, z. B. besitzt eine E-Mail einen Absender. Ein weiteres Datenelement sind beispielsweise die AGBs. Diese besitzen u.a. die für die Rechtsverträglichkeit relevante die Eigenschaft „schriftliche Zustimmung“.

Funktionselemente sind die Aktionen, die der Nutzer mit den Datenelementen und ihren Repräsentationen in der Benutzungsschnittstelle durchführen kann. Das sind beispielsweise das Suchen, Erstellen, Bearbeiten, Löschen und Speichern von Datenelementen. Die AGBs besitzen z. B. das Funktionselement Akzeptieren.

Ausgangspunkt für die Definition der Daten- und Funktionselemente ist das Anforderungsdokument. Jedes Daten- und Funktionselement muss auf eine Anforderung zurückführbar sein, wodurch gewährleistet wird, dass jedes Element einen wohldefinierten Zweck erfüllt. Eine Anforderung kann jedoch auch mehrere Daten- und Funktionselemente erforderlich machen. Bei der Definition und Benennung der Elemente sollten die mentalen Modelle des Nutzers berücksichtigt werden. Design Patterns und Design-Prinzipien finden in diesem Vorgehensbaustein Einsatz, um bereits vorhandenes Design-Wissen wiederzuverwenden und dadurch diese Teilaktivität effizienter zu bewältigen. Beispielsweise sollten Standards eingehalten werden. Außerdem sollten etwa die Datenelemente „E-Mails“ den üblichen mentalen Modellen der Nutzer folgen und die übliche Benennung und Attribute aufweisen sowie übliche Funktionen damit möglich sein.

Nach der Definition einer größeren Anzahl an Daten- und Funktionselementen werden diese in Beziehung zueinander gebracht und damit in funktionale Einheiten gepackt und ihre Hierarchie zueinander festgestellt. Dabei sollte die Gruppierung der Elemente den Handlungsablauf wie er in den Anwendungsfällen beschrieben wurde bzw. wie er zwischen diesen Anwendungsfällen erforderlich ist möglichst gut unterstützen.

Folgende Fragen sollten dabei berücksichtigt werden:

- Welche Elemente benötigen eine große Menge an Raum und welche nicht?
- Welche Elemente sind Container für andere Elemente?
- Wie sollten die Container angeordnet werden, um den Handlungsablauf zu unterstützen?
- Welche Elemente werden zusammen verwendet und welche nicht?
- In welcher Reihenfolge wird eine bestimmte Menge an verknüpften Elementen verwendet?
- Welche Design Patterns und Design-Prinzipien sind anzuwenden?
- In welcher Weise haben die mentalen Modelle des Nutzers einen Einfluss auf die Anordnung?

Die Gruppierung der Daten- und Funktionselemente endet damit, diese ansatzweise in Einzelansichten der Benutzungsschnittstelle, welche als Container zu betrachten sind, zu sammeln. Obgleich sich diese Gruppierung im Verlauf der weiteren Vorgehensbausteine ändern kann, ist sie nützlich um das Konzeptdesign zu beschleunigen. Die dabei gefundenen Hauptansichten der Benutzungsschnittstelle bündeln miteinander verknüpfte Anforderungen.

Beispiel für Daten und Funktionselemente

In der folgenden Tabelle werden exemplarische Datenelemente und einige zugehörige Funktionselemente für die Benutzungsschnittstelle von Meet-U sowie ihr Ursprung – entweder Anforderung oder Anwendungsfall – dargestellt.

Ursprung für das Element (zitierte Anforderung oder zitierter Ausschnitt aus einem Anwendungsfall)	**Resultierendes Datenelement**	**Beispielhafte zugehörige Funktionselemente**
“Der Nutzer soll jederzeit sein persönliches Profil aktualisieren können. Der Nutzer kann seine persönlichen Präferenzen und deren Bewertungen eintragen. Beispielsweise für Kino, Sport, Theater oder Konzert, Vorlieben hinsichtlich Events, die von ihm bevorzugten Verkehrsmittel, seine Lieblingsmusik oder bevorzugte Filmgenres.”	Nutzerprofil	Bearbeiten des Profilnamens, Bearbeiten der Interessen, Bearbeiten der Verkehrsmittel-Präferenz
“Der Nutzer ruft den Dialog zum Erstellen eines Events auf.”	Events (Veranstaltungen),	Suchen von öffentlichen Events, Erstellen eines privaten Events
“Das System zeigt dem Nutzer an, dass eine neue Einladung vorliegt.”, “Meet-U zeigt eine Liste mit ausstehenden Einladungen an.”, “Der Nutzer wählt die Einladung aus, die er bestätigen möchte.”	Einladungen zu Events	Annehmen einer Einladung, Ablehnen einer Einladung

“Der Nutzer soll Kontakte zu einem privaten Event einladen können.”	Kontakte, Kontaktliste	Auswahl eines Kontaktes aus einer Liste
“Der Nutzer wird darauf aufmerksam gemacht, in welcher Form er für den Dienst bezahlt (einmalige Zahlung von 3,99 Euro oder kostenlose Variante mit Schaltung personenbezogener Werbung und Verkauf von Daten). Der Nutzer muss die AGB von Meet-U bestätigen/einsehen können.”	AGBs	Bestätigen der AGBs, Ablehnen der AGBs

Tabelle 4: Exemplarische Daten- und Funktionselemente für die Benutzungsschnittstelle von Meet-U

4.1.4 Aktivität: Workflows erarbeiten

Ziel der Erarbeitung von Workflows ist die Definition der Informationsarchitektur und Funktionsstruktur der Benutzungsschnittstelle. Ein Workflow stellt alle für das Erreichen einer Aufgabe notwendigen Ansichten der Benutzungsschnittstelle und die möglichen Übergänge dazwischen dar. Grundsätzlich soll die Gebrauchstauglichkeit der Benutzungsschnittstelle optimal unterstützt werden. Allerdings kann die Gewährleistung gleichzeitiger Rechtsverträglichkeit hierbei Konflikte erzeugen, z. B. indem gefordert wird, dass das Impressum von jeder Stelle in der Benutzungsschnittstelle über ein bis maximal zwei Klicks erreichbar sein soll.

Als Hilfsmittel zur Erarbeitung von Workflows dienen grafische Modellierungssprachen für Programmabläufe wie beispielsweise die DIN 66001 [92]. Die Erarbeitung kann einerseits mit Papier und Bleistift erfolgen, andererseits können aber auch Software-Werkzeuge wie beispielsweise Microsoft Visio genutzt werden.

Ein Workflow zeigt den Interaktionsablauf zwischen Nutzer und Benutzungsschnittstelle zur Bewältigung einer bestimmten Aufgabe, die mit einem Anwendungsfall (siehe Abschnitt 4.1.2) definiert wurde. Jeder Anwendungsfall sollte mindestens in Form eines Workflows grafisch modelliert werden. Für essentielle Anwendungsfälle können unter Umständen mehrere Workflows notwendig werden, um alle wichtigen Ausnahmebedienungen in einem ausreichenden Detailgrad zu modellieren. Eine Beschreibung von Workflows, ihrer Erstellung und ihre Bedeutung für das Konzeptdesign findet sich bei Stapelkamp [93].

Ein Workflow zeigt jeden einzelnen Interaktionsschritt für das Erreichen einer bestimmten Aufgabe. Inbegriffen sind hierbei jede Eingabe des Nutzers und jede Ausgabe der Benutzungsschnittstelle. Es werden zudem notwendige und optionale Abläufe unterschieden.

Für die Erstellung des Workflows sind die Elemente der Modellierungssprache einheitlich zu verwenden. Einen Auszug aus der Modellierungssprache DIN 66001 [92], angepasst für die Modellierung der Workflows, ist in Tabelle 5 dargestellt.

	Start / Ende eines Workflows
	eine einzelne Ansicht der Benutzungsschnittstelle im Workflow
	Dateneingabe durch den Nutzer
	Richtung des Interaktionsflusses

Tabelle 5: Auszug aus der Modellierungssprache DIN 66001 [92]

Der Workflow enthält neben der Darstellung der einzelnen Schritte auch Details darüber, welche Informationen die Ansichten darstellen und welche konkreten Operationen der Nutzer für den Übergang zur nächsten Ansicht ausführen kann oder muss.

Die Erstellung der Workflows kann durch Design-Prinzipien und Design Patterns effizienter werden. Das Design-Prinzip „Konsistenz“ [94] empfiehlt z. B., dass Aufgaben der gleichen Art durch die Ausführung gleicher Interaktionen mit der Benutzungsschnittstelle vom Nutzer bewältigt werden können. Das Design Pattern „Wizard“ [95] beschreibt z. B. die geleitete Bearbeitung einer dem Nutzer unbekannten Aufgabe.

Die Modellierung von Workflows basiert auf der zeitlich-logischen Reihenfolge der Interaktionsschritte zur Erledigung der bestimmten Aufgabe und bildet die Dialoggestaltung der Benutzungsschnittstelle. Ein Workflow stellt also nur einen Ausschnitt aus den Möglichkeiten in der Verwendung der Benutzungsschnittstelle dar.

Mit Hilfe von Workflows wird auch das Dialogverhalten durch die *Adaptionen* eines ubiquitären Systems definiert. Denn Adaptionen, die eine Auswirkung auf die Benutzungsschnittstelle haben, erzeugen eine Dialogsituation, da der Nutzer versucht diese Änderungen zu interpretieren, zu verstehen und darauf zu reagieren. Ist eine Adaptation jedoch nicht ausreichend transparent oder steuerbar, so kann dies Verwirrungen, Aufgabenunterbrechungen oder gar Aufgabenabbruch zur Folge haben [96], [97], [98], [99]. Ein Gefühl von Kontrollverlust gilt es im Sinne der Sozialverträglichkeit zu vermeiden.

Transparenz ist das Ausmaß zu dem der Nutzer die Aktionen des Systems verstehen kann und zu dem er ein Verständnis hat, wie das System funktioniert [91]. Steuerbarkeit ist das Ausmaß in dem der Nutzer die Möglichkeit hat bestimmte Systemstatus zu ermöglichen oder zu verhindern, sofern er die Absicht dazu hat [91].

Um transparente und steuerbare Adaptionen zu erhalten, wurde im Rahmen der VENUS-Entwicklungsmethode ein benachrichtigungsbasierter Ansatz entwickelt [100]. Benachrichtigungen informieren normalerweise über Dinge, Vorgänge usw., mit denen der Nutzer aktuell nicht beschäftigt ist [101]. Mit Hilfe einer Benachrichtigung kann die Steuerbarkeit einer Adaptation ermöglicht werden, weshalb eine jede Adaptation durch eine Benachrichtigung begleitet wird. Zwei grobe Möglichkeiten für die Steuerbarkeit der Adaptation:

Vorher-Steuerbarkeit: Die Steuerbarkeit wird ermöglicht bevor die Adaptation durchgeführt wird. Der Nutzer wird gefragt, ob der die Adaptation akzeptiert, sie ablehnt oder sie vertagen will. Nachdem der Nutzer die Adaptation akzeptiert, wird die Adaptation durchgeführt und ein Dienst o.ä. angezeigt (siehe Abbildung 11 a).

Nachher-Steuerbarkeit: Die Steuerbarkeit wird ermöglicht, nachdem die Adaptation bereits durchgeführt wurde und der Dienst bereits läuft. Der Nutzer kann die Adaptation rückgängig machen und damit den laufenden Dienst schließen (siehe Abbildung 11 b).

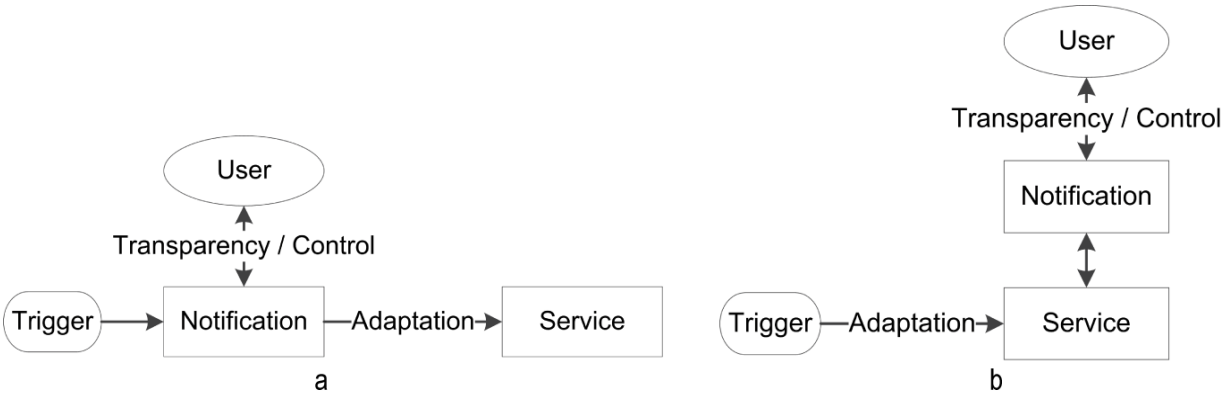


Abbildung 11: Dialogschritte für Vorher-Steuerbarkeit (a) und Nachher-Steuerbarkeit (b).

Beispiel für Workflows

Der Anwendungsfall „Anmeldung“ wurde im Workflows siehe Abbildung 11 modelliert. Nach dem Start von Meet-U bekommt der Nutzer eine Anmelde-Ansicht zu sehen. Dort kann er seinen Account-Namen und Passwort eingeben, um dann zum Dashboard (Hauptbildschirm von Meet-U) zu gelangen. Gibt er einen falschen Nutzernamen oder ein falsches Passwort ein, wird er gefragt, ob er schon als Nutzer registriert ist. Wird dies mit Nein beantworten, wird der Nutzer zur Registrierung veranlasst. Hierfür existiert es einen separaten Anwendungsfall bzw. Workflow.

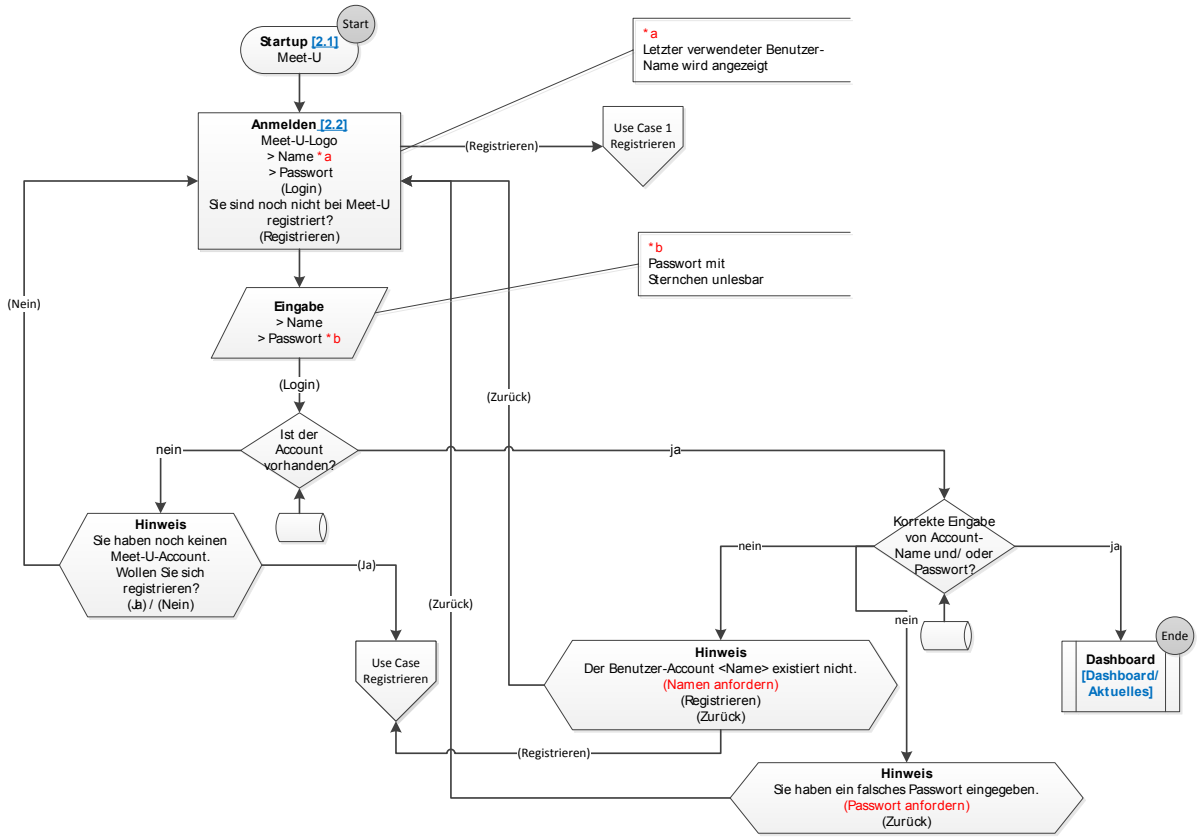


Abbildung 12: Workflow für den Anwendungsfall Anmelden des Nutzers in Meet-U

4.1.5 Aktivität: Sitemaps erarbeiten

Ziel einer Sitemap ist die Ergänzung von bestimmten Workflows. Sie wird ebenso grafisch modelliert. Eine Sitemap stellt das Grundgerüst der Benutzungsschnittstelle in einer hierarchischen Form dar, indem sie die wichtigsten Ansichten und wie diese zueinander in absoluter Beziehung stehen darstellt. Damit erleichtert die Sitemap die Einordnung der Workflows.

Grundlage für die Sitemap bilden die Daten- und Funktionselemente (siehe Abschnitt 3.3) und die Workflows (siehe Abschnitt 4.1.4). In der Sitemap werden die einzelnen Bereiche der Anwendung und ihre hierarchischen oder inhaltlich-logischen Zusammenhänge grafisch modelliert. Eine Sitemap enthält keine Informationen darüber, welche Operationen zu einem Übergang von einem Bereich der Anwendung zu einem anderen führen. Die Sitemap stammt ursprünglich aus dem Webdesign. Eine Beschreibung von Sitemaps findet sich zum Beispiel bei [102]. Im Konzeptdesign für ein System sollte mindestens eine Sitemap erarbeitet werden.

Für eine Sitemap können ausgewählte Elemente aus einer grafischen Modellierungssprache wie der DIN 66001 [92] verwendet werden, wie die Tabelle 6 beispielhaft zeigt, da die Sitemap weniger komplex ist.

	eine einzelne Ansicht der Benutzungsschnittstelle in der Sitemap
	Richtung der hierarchischen Zusammenhänge

Tabelle 6: Beispielhafte Elemente einer Sitemap

Beispiel für eine Sitemap

In der Sitemap in Abbildung 13 sind die wichtigsten Hauptansichten von Meet-U grafisch modelliert. Der Hauptbildschirm, bzw. auch Dashboard genannt, stellt den zentralen Einstiegspunkt in der Anwendung dar, von der jegliche Hauptansichten zugänglich sind.

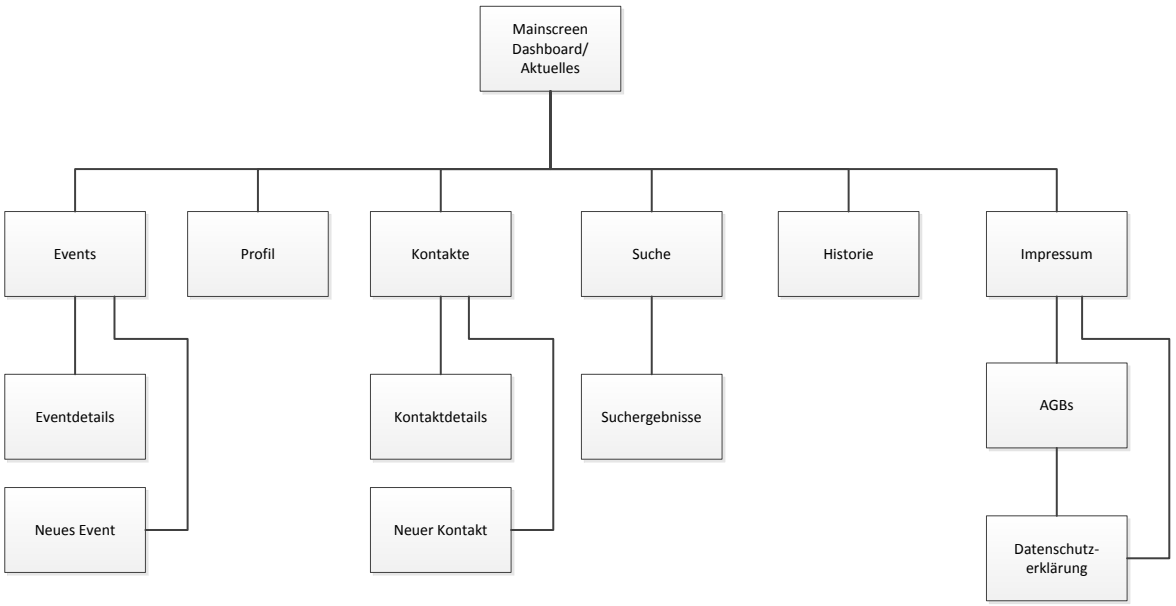


Abbildung 13: Überblicks-Sitemap für Meet-U

4.1.6 Aktivität: Erarbeitung der Funktionslayouts

Zunächst werden die Informationen und Funktionselemente, die in den Ansichten der Workflows nur textlich benannt sind, grafisch auf der jeweiligen Ansicht angeordnet. Dies wird als Funktionslayout bzw. auch Wireframe [93] bezeichnet.

Grundsätzlich sollten die Haupt-Ansichten zuerst erarbeitet werden. Als Haupt-Ansichten sind diejenigen Ansichten anzusehen, mit denen der Nutzer hauptsächlich bei der Interaktion konfrontiert wird. Der Hauptbildschirm ist beispielsweise eine Haupt-Ansicht, da er den Ausgangspunkt für alle Interaktionen mit der Benutzungsschnittstelle darstellt. Das Funktionslayout der Haupt-Ansichten sollte maßgebend für die weiteren zu erarbeitenden Funktionslayouts anderer Ansichten sein, um eine konsistente Anordnung gleicher Elemente auf den verschiedenen Ansichten der Benutzungsschnittstelle zu gewährleisten.

User Interface Design Patterns oder Usability-Design-Prinzipien können in dieser Teilaktivität eingesetzt werden, um sie effizienter durchzuführen. User Interface Design Patterns beschreiben bewährte Gestaltungslösungen für eine Benutzungsschnittstelle. Sie beinhalten Details zum Funktionslayout der notwendigen Ansichten und Details zur Interaktionsgestaltung. Beispielsweise

beschreibt das User Interface Design Pattern „Carousel“ [95] wie das Funktionslayout für eine Ansicht, die eine Menge an für den Nutzer interessanten Datenelementen, wie z. B. Fotos, gebrauchstauglich anordnet und zugänglich macht.

Usability-Design-Prinzipien sind übergreifende Gestaltungsempfehlungen zur Gewährleistung der Gebrauchstauglichkeit der Benutzungsschnittstelle. Beispielsweise beschreibt das Prinzip „Konsistenz“ [94], dass das Funktionslayout für Ansichten ähnlichen Zwecks auch gleich gestaltet sein sollte. Dies dient der Erwartungskonformität der Benutzungsschnittstelle. Beispielsweise sollte eine Funktion, die aus jeder Ansicht erreichbar sein muss (wie beispielsweise die Beenden-Funktion) über das gleiche Icon, das gleich positioniert ist, erreichbar sein.

4.1.7 Aktivität: Erarbeitung der visuellen Gestaltung

In dieser Aktivität erfolgt die Gestaltung des konkreten Erscheinungsbilds der Benutzungsschnittstelle, in dem die Elemente der Funktionslayouts eine visuelle Gestaltung erhalten. Diese Teilaktivität umfasst die Detailgestaltung der visuelle, die auditive und die haptische Ebenen. Die zu gestaltenden Aspekte auf der visuellen Ebene sind:

- die Größe einzelner Elemente,
- die Formgestaltung,
- die Farbwahl,
- die Gestaltung des Hintergrunds,
- die gewählten Schriftarten und Schriftschnitte
- sowie die Helligkeit und Kontrast.

Auch in dieser Teilaktivität kommen Usability-Design-Prinzipien zum Einsatz. Beispielsweise liefert die DIN 9241-12 [42] Vorgaben zur Gestaltung von unterschiedlichen Funktionsfeldern, um diese visuell unterscheidbar zu machen. Ebenso werden Icons in dieser Teilaktivität ausgearbeitet. Ein Icon abstrahiert und visualisiert den Informationsgehaltes eines Objekts, eines Themas, einer Funktion oder Benutzungsbeschreibung. In Form eines Wortes oder Textes würde eine Beschreibung erheblich mehr Platz in Anspruch nehmen und könnte auch nicht so schnell wahrgenommen werden wie ein Icon. Sofern Textelemente vermieden werden, lassen sich mit Icons zudem kulturelle und sprachliche Schranken überwinden [93]. Zur Signalisierung von Zuständen bzw. Fehlern, wird der erklingende Ton, Klang oder das Geräusch, in Anlehnung an das Wort Icon, auch als Earcon bezeichnet. Diese werden ebenso in dieser Teilaktivität gestaltet. Audiodesign-Tools dienen der Entwicklung der Earcons und anderer Audio-Signale. Weitere Werkzeuge für diesen Vorgehensbaustein zur Erarbeitung der visuellen Gestaltung sind Prototyping-Tools oder Grafikbearbeitungsprogramme.

Die Transparenz der *Adaptionen* wird im Rahmen der Erarbeitung der Funktionslayouts definiert. Wie bereits bei der Aktivität Workflows beschrieben wurde, sind transparente und steuerbare Adaptionen das Ziel im Rahmen der VENUS-Entwicklungsmethode. Transparenz ist das Ausmaß zu dem der Nutzer die Aktionen des Systems verstehen kann und zu dem er ein Verständnis hat, wie das System funktioniert [91]. Je mehr Platz der Benachrichtigung zur Verfügung steht, desto mehr Informationen können dem Nutzer übermittelt werden. Basierend auf dem Design Pattern Benachrichtigungen von Hoober & Berkman [103] können drei Möglichkeiten ausgewählt werden um ein verschiedenen Maß an Transparenz zu ermöglichen (siehe Abbildung 14).

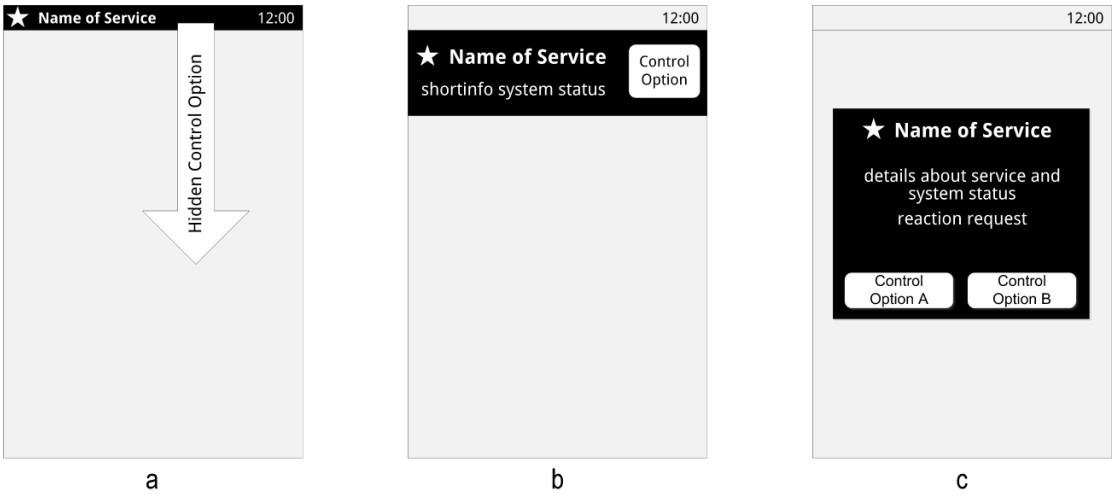


Abbildung 14: Schematische Darstellung für drei Benachrichtigungsoptionen, die unterschiedlichen Raum zur Verfügung stellen, um unterschiedliche Transparenzbedürfnisse zu berücksichtigen: Annunciator Row (a), Notification Strip (b) und Pop-Up (c).

Beispiel für visuelle Gestaltung

Die Registrierung des Nutzers bei Meet-U bedarf drei relevanter Ansichten. Die Detailgestaltung der Ansichten ist in Abbildung 15 dargestellt.

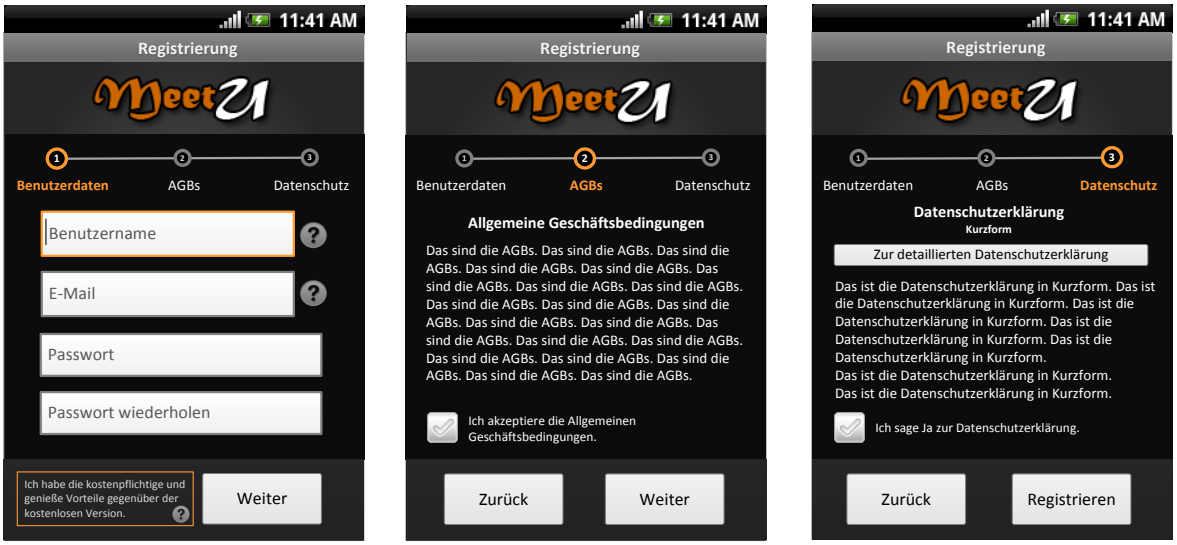


Abbildung 15: High-Fidelity-Entwürfe für die drei wichtigsten Ansichten der Benutzungsschnittstelle aus dem Anwendungsfall Registrieren

4.2 Werkzeugunterstützung: Interdisziplinäre Entwurfsmuster

Die interdisziplinären Entwurfsmuster wirken unterstützend bei der Anwendung der VENUS-Entwicklungsmethode. Sie werden aus Projekten gewonnen, in denen die VENUS-Entwicklungsmethode bereits erfolgreich zur Anwendung kam. Im Besonderen benennen sie normative Aspekte, die in bestimmten Szenarien zu beachten sind, und führen für diese konkrete Umsetzungs- und Lösungsmöglichkeiten an. Dadurch können sie in neuen Projekten als Leitfaden und Ideengeber zum Einsatz kommen. Tabelle 7 erläutert die Struktur interdisziplinärer Entwurfsmuster.

Name	Der Name des Entwurfsmusters ist derart gewählt, dass er mit seinem Zweck einfach in Verbindung gebracht werden kann und gleichzeitig leicht zu merken ist. Dies beschleunigt die Kommunikation unter den Entwicklern und zwischen den Disziplinen.
Zweck	Es wird das Problem umrissen, welches sich das Entwurfsmuster annimmt. Das dargestellte Problem ist allerdings kein technisches, sondern ein abstraktes und nichtfunktionales, welches seinen Ursprung in den normativen Anforderungen findet. In ein oder zwei Sätzen wird ebenso zusammengefasst, wie das Problem angegangen wird. Dies unterstützt den Anwender des Musterkataloges beim Überfliegen der Muster. Die Beschreibung des Zwecks sollte fünf Sätze nicht überschreiten.
Motivation	Das Problem wird anhand eines Beispiels veranschaulicht. Das Beispiel muss repräsentativ für die Situationen und Szenarien sein, in welchen das Muster zum Einsatz kommen soll.
Kontext und wirkende Kräfte	Normative und nichtfunktionale Aspekte, welche vom Entwurfsmuster adressiert werden, werden hier aufgelistet und in wenigen Worten beschrieben. Im Besonderen wird auf in Konflikt stehende Ziele eingegangen, etwa wenn normative Anforderungen im Widerspruch zu funktionalen Anforderungen stehen. Das Hauptziel von Entwurfsmustern ist es diese Konflikte so gut wie möglich zu lösen.
Lösung	Der Lösungsteil ist in zwei Abschnitte aufgeteilt. Der erste Abschnitt beschreibt die Idee und das allgemeine Vorgehen und ist nicht auf ein bestimmtes Szenario festgelegt. Der Detailgrad der Lösung ermöglicht das Herausarbeiten konkreter Entwicklungsziele, ist aber nicht derart präzise, dass es sich auf ein Anwendungsgebiet beschränkt. Der zweite Teil der Lösung stellt beispielhaft den Einsatz des Entwicklungsmusters in erfolgreich umgesetzten Anwendungen vor.
Konsequenzen	Sowohl positive als auch negative Konsequenzen, welche sich durch Anwendung des Entwurfsmusters ergeben können, werden hier zusammengefasst. Diese sind zumeist nichtfunktionaler Natur, können allerdings auch technischen Ursprungs sein, sollten sie von zentraler Bedeutung für das Entwicklungsteam sein.
Verwandte Anforderungsmuster	Anforderungsmuster, welche in Beziehung zum Entwurfsmuster stehen, werden hier gelistet, jedoch nicht weiter ausgeführt.
Verwandte HCI-Muster	Die HCI-Muster, auf welche hier verwiesen wird, sollen die Entwickler bei der Umsetzung des Entwurfsmusters unterstützen.

Tabelle 7: Aufbau eines interdisziplinären Entwurfsmusters

Während Anforderungsmuster die Identifizierung und Dokumentation von Anforderungen unterstützen, haben interdisziplinäre Entwurfsmuster zum Ziel, funktionale Anforderungen derart umzusetzen, dass sie möglichst in Einklang mit normativen Richtlinien und Anforderungen stehen. Dazu wird die vorgeschlagene technische Lösung aus Sicht von Recht, Vertrauen und Usability gerechtfertigt.

Im Projekt erstellte Anforderungsmuster sind [104]:

- Enable/Disable Functions
- Trust and Transparency
- Abridged Terms and Conditions
- Context State Indication
- Control of Autonomous Adaptation
- Emergency Button
- Data Access Log
- On Demand Explanation

Zusammenfassend vereinfachen und beschleunigen die Entwurfsmuster den interdisziplinären Entwicklungsprozess. Sie erleichtern zudem die Kommunikation zwischen den verschiedenen beteiligten Disziplinen durch Darstellung konkreter Szenarien und Benennung der Schnittpunkte zwischen den Disziplinen. Die Liste der Entwurfsmuster ist jedoch keinesfalls vollständig oder umfassend. Sie bedarf der ständigen Erweiterung und Anpassung und fordert Entwickler als auch beteiligte Disziplinen dazu auf, mühsam und gemeinsam erarbeitete Lösungen als Muster festzuhalten.

5 Softwaredesign / Implementierung (Phase 4)

Besonderheiten der Phase Softwaredesign/Implementierung in der VENUS-Entwicklungsmethode:

- Integration verschiedener etablierter Vorgehensmodelle der Softwareentwicklung durch Konnektoren
- Kontinuierliche Expertenvalidierungen bezüglich (normativer) Anforderungen

Im Softwaredesign und der Implementierung schreibt die VENUS-Entwicklungsmethode kein konkretes Vorgehen vor, sonders versucht es den Entwicklungsteams zu ermöglichen, ihr gewohntes Vorgehen zu verwenden. Jedoch verlangt die VENUS-Entwicklungsmethode regelmäßige Expertenvalidierungen bezüglich der Umsetzung der Anforderungen. Somit müssen diese Überprüfungen während der Umsetzung ermöglicht werden. Zur leichteren Vereinbarung der VENUS-Entwicklungsmethode und verschiedenen Vorgehensmodellen werden hier Konnektoren vorgestellt. Bisher existieren Konnektoren für die Modellgetriebene Softwareentwicklung, Extreme Programming und Scrum. Diese beschreiben, wie die Einbindung einzelner Vorgehensmodelle in die VENUS-Entwicklungsmethode erfolgen kann. Diese Überlegungen lassen sich jedoch auch auf andere Vorgehensmodelle übertragen.

Bei den Konnektoren werden drei Verbindungspunkte als Aktivitäten im Detail erläutert:

- Vorbereitung
- Prototypen erstellen
- Validierungsergebnisse einarbeiten

Bei der ersten Teilaktivität werden nötige Vorarbeiten beschrieben, um die entstehenden Produkte der VENUS-Entwicklungsmethode für den Entwicklungsprozess vorzubereiten. In der zweiten Teilaktivität wird erläutert wie im laufenden Entwicklungsprozess ein Prototyp für die Expertenvalidierungen erstellt werden kann. In der letzten Teilaktivität wird beschrieben, wie die Resultate aus den Expertenvalidierungen wieder in den laufenden Entwicklungsprozess zurückfließen.

5.1 VB: Konnektor Modellgetriebene Softwareentwicklung

Produkt	Applikation, Prototypen
Verantwortliche Rolle	System Engineer
Mitwirkende Rollen	

Dieser Vorgehensbaustein beschreibt, wie die Modellgetriebene Softwareentwicklung (MDSD) als Implementierungsmethode in der VENUS-Entwicklungsmethode genutzt werden kann. Hierbei wird folgende Definition für MDSD zugrunde gelegt: „Modellgetriebene Softwareentwicklung (Model Driven Software Development, MDSD) ist ein Oberbegriff für Techniken, die aus formalen Modellen automatisiert lauffähige Software erzeugen“ [105].

Insgesamt lassen sich in der Definition drei Bereiche identifizieren. Der erste Bereich ist die Nutzung eines formalen Modells um eindeutig das gewünschte Verhalten des zu entwickelnden Systems zu beschreiben. Häufig werden hierbei UML-Diagramme verwendet, jedoch sind nicht unbedingt grafisch repräsentierbare Modelle notwendig. Alternativ lässt sich ebenfalls ein textuelles Modell nutzen. Formales Modell bedeutet an dieser Stelle, dass das Modell möglichst viele Aspekte der Software vollständig und eindeutig beschreiben kann.

Der nächste Schritt ist, aus diesem Model lauffähige Software zu erzeugen. Eine solche Transformation kann sowohl durch ein Werkzeug wie auch von Hand durchgeführt werden. Eine manuelle Transformation, also die Interpretation des Modells und Umwandlung in Quellcode durch einen Entwickler, ist jedoch nicht sinnvoll, da die Korrektheit nicht sichergestellt werden kann. Hinzu kommt, dass das Modell nicht nur zur Dokumentation dient, spätere Änderungen sollen direkt im Modell vorgenommen werden. Dies erfordert die Gewährleistung einfacher Verwendbarkeit für den Entwickler. Somit ist eine werkzeuggestützte Transformation vorzuziehen. Es werden entsprechende Werkzeuge benötigt, um mit dem Modell zu arbeiten und die Transformation automatisch durchzuführen.

In [106] wird ein iterativer MDSD Ansatz beschrieben. Dieser verwendet eine domänenspezifische Sprache (Domain Specific Language, DSL), welche die durch die Domäne vorgegebenen Richtlinien, Einschränkungen und auch Möglichkeiten abbildet. Es ist Ziel spezielle Probleme aus dieser Domäne zu erfassen und zu beschreiben um einen hohen Grad an Anpassung zu erreichen. Das Modell der Anwendung wird mit den Sprachmitteln der DSL beschrieben. Das in der DSL repräsentierte Wissen erleichtert die Entwicklung einer Anwendung, da komplexe Zusammenhänge durch simple Sprachkonstrukte modelliert werden können. Eine allgemein für die Domain erzeugte Sprache kann für weitere Projekte in derselben Domäne wiederverwendet werden. Existiert noch keine DSL, so muss diese zuvor oder parallel mit der Anwendung entwickelt werden. Alternativ kann auch eine allgemeine Sprache wie UML ohne domainspezifische Anpassungen genutzt werden. Somit entfällt der Aufwand eine DSL zu entwickeln, jedoch erhöht sich damit der programmieraufwand.

5.1.1 Aktivität: Vorbereitung

In der Vorbereitungsphase werden die Produkte der VENUS-Entwicklungsmethode auf die notwendigen Produkte des MDSD-Ansatzes überführt. Hierbei gibt es drei Fälle zu unterschieden:

- Es existiert bereits eine DSL, welche für das neue Produkt genutzt werden kann
- Es existiert keine DSL und diese muss mit dem Produkt entwickelt werden
- Es existiert keine DSL und es soll eine Standardsprache genutzt werden

Existiert bereits eine DSL, welche für das neu zu entwickelnde System verwendet werden kann, muss in der Vorbereitungsphase lediglich überprüft werden, ob die in der DSL vorgesehenen sozialverträglichen Komponenten auch auf das aktuelle System passen. Hierfür dient die Anforderungssammlung aus Phase 2 (vgl. Abschnitt 3) als Grundlage. Zunächst werden alle Anforderungen extrahiert, welche einen sozialverträglichen Hintergrund haben. Anschließend überprüft der Systemarchitekt, welche dieser Anforderungen bereits mit der DSL abgedeckt sind. Wichtig ist hierbei nicht nur zu prüfen, ob eine Anforderungen umgesetzt ist, sondern auch zu prüfen, ob die Umsetzung mit den anderen Anforderungen übereinstimmt. Wiedersprüche und Unklarheiten sollten in einem Workshop mit den Experten besprochen werden. Hierbei stellt der Systemarchitekt die unklaren/widersprüchlichen Punkte vor und erläutert diese mit den Experten.

Ist noch keine DSL für die Domäne der Anwendung vorhanden, so kann entweder parallel zur Anwendung eine DSL entwickelt werden oder es wird eine allgemeine Beschreibungssprache wie UML verwendet. Um eine neue DSL zu entwickeln wird eine Anforderungssammlung benötigt. Hierbei die in Abschnitt 3 erstellte Anforderungssammlung ohne Anpassungen als Grundlage verwendet werden. Da in der Anforderungssammlung bereits die Anforderungen der verschiedenen Experten vereinheitlicht wurden, können alle gleich abgearbeitet werden. Zusätzlich ermöglicht die Neuentwicklung einer DSL sozialverträgliche Anforderungen direkt zu verankern. Die Entwicklung wird iterativ durchgeführt, um den zusätzlichen Aufwand möglichst gering zu halten. Es ist nicht nötig, dass bereits alle angedachten Sprachelemente von Beginn an bereits vorhanden sind. Es empfiehlt sich mit Elementen zu beginnen, die auch für die zuerst umzusetzenden Anwendungsfunktionalitäten benötigt werden. Die Versionen der DSL können, ähnlich wie ein Prototyp, von fachkundigen Experten bereits vor der Nutzung in der Anwendungsentwicklung validiert werden (vgl. Abschnitt 5.1.3). Ein weiterer Vorteil der VENUS-Entwicklungsmethode ist die Verfügbarkeit der Experten, welche bei Rückfragen und Unklarheiten von den SW-Entwicklern befragt werden können. Die Entwicklung kann bereits parallel zum Konzeptdesign beginnen, da zu diesem Zeitpunkt die Anforderungssammlung bereits existiert und die Produkte aus dem Konzeptdesign keinen direkten Einfluss auf die DSL haben.

Wird keine domänenspezifische Sprache verwendet, so ist keine weitere Vorarbeit nötig und es kann direkt mit der Umsetzung der Anwendung begonnen werden.

5.1.2 Aktivität: Prototypen erstellen

Für die softwareentwicklungsbegleitende Validierung ist es nötig, in regelmäßigen Abständen einen Prototyp zu erstellen. Dieser wird verwendet, um die korrekte Umsetzung der Anforderungen zu überprüfen. So können nicht nur Fehler in der Implementierung, sondern auch fehlende oder unnötige Anforderungen rechtzeitig gefunden werden. Hierfür werden zyklisch Expertenvalidierungen durchgeführt (vgl. Abschnitt 5.4). Die Dauer eines Zyklus muss individuell von Projekt zu Projekt entschieden werden. Ausschlaggebend für die Länge eines Zyklus ist unter anderem die Verfügbarkeit der Experten. Bei einer geringeren Verfügbarkeit empfiehlt sich ein längerer Zyklus von etwa

vier bis sechs Wochen. Bei einer hohen Verfügbarkeit empfiehlt sich ein Zyklus von etwa zwei Wochen.

5.1.3 Aktivität: Validierungsergebnisse einarbeiten

Bei der entwicklungsbegleitenden Validierung werden in der VENUS-Entwicklungsmethode, neben den Tests und Qualitätskontrollen, weitere Expertenvalidierungen durchgeführt. Hierfür werden die Prototypen aus dem vorherigen Schritt von den Experten begutachtet und entsprechende Änderungen können so frühzeitig in den Entwicklungsprozess zurückfließen. Dies geschieht mittels einer von den Experten erstellen Liste mit neuen oder angepassten Anforderungen. Diese werden vom Systemarchitekten entgegengenommen und in die ursprüngliche Anforderungssammlung integriert.

Neben diesen Expertenvalidierungen der Prototypen kann bei der Verwendung einer DSL auch diese durch die Experten validiert werden. Hierbei ist zu bedenken, dass die Experten nicht mit formalen Modellen vertraut sind. Es ist daher nicht praktikabel lediglich die DSL mit entsprechenden Erklärungen, analog zu den Prototypen, den Experten zu Verfügung zu stellen. Es wird empfohlen, zumindest die ersten Evaluierungen im Rahmen von Workshops durchzuführen. Hierbei präsentiert der Systemarchitekt die für die Experten relevanten Sprachelemente und erläutert die daraus resultierende Umsetzung. Die Experten bewerten diese und äußern Verbesserungen und Einwände. Diese Workshops helfen ein beidseitiges und besseres Verständnis für die DSL zu entwickeln. Spätere kleinere Änderungen sollten aus zeitlichen Gründen nicht mehr in Workshops validiert werden. Es wird empfohlen diese bilateral mit dem oder den betroffenen Experten direkt zu klären.

5.2 VB: Konnektor Xtreme Programming (XP)

Produkt	Applikation, Prototypen
Verantwortliche Rolle	System Engineer
Mitwirkende Rollen	

Extreme Programming XP wurde in den 1990er Jahren von Kent Beck vorgestellt. Es ist mit den sogenannten agilen Methoden verwandt und ist ein sehr häufig verwendeter Ansatz zur Softwareerstellung. Die Grundidee ist die schnelle Erstellung einfacher lauffähiger Software die im Laufe des XP Prozesses immer weiter verfeinert wird, bis sie dem vom Kunden gewünschten Funktionsumfang entspricht. Dabei wird besonderes Augenmerk auf eine häufige Ausleitung eines Prototypen und einer anschließenden Überprüfung seiner Funktion gelegt. Die Ergebnisse der Überprüfung werden dann zur Verbesserung der Software benutzt. Dieser Prozess wird viele Male zyklisch durchlaufen [107], [108].

Die Wahl von XP zum Erstellen der eigentlichen Software führt zu einer Entwicklung der Software in kleinen Bausteinen mit häufigem Feedback bis zur hinreichenden, das heißt den Kunden zufriedenstellenden, Anpassung an die Anforderungsvorgaben. Die Schritte einer Iteration im XP lauten:

1. Risikoanalyse und Nutzenanalyse,
2. Prototyping und
3. Akzeptanztest.

5.2.1 Aktivität: Vorbereitung

In der ersten Aktivität werden die sogenannten User Stories festgehalten. Dabei handelt es sich um vereinfachte Use Cases [107]. Inhalt einer User Story ist der Ablauf eines Prozesses in der realen Welt, in dem die Software eingesetzt werden soll. Sie werden von dem Kunden in zwei bis drei einfachen Sätzen geschrieben und enthalten die Terminologie des Kunden. Die anderen Aspekte der VENUS-Entwicklungsmethode werden im nächsten Schritt eingebracht. Dafür werden die Anforderungen zur sozialverträglichen Softwaregestaltung in einer Neuformulierung der User Stories mit eingebaut und sind danach nichtmehr von denen des Kunden zu unterscheiden. Dafür ist es hilfreich, wenn die Anforderungen aus diesen Bereichen bereits vorliegen. Wenn die Anforderungen noch nicht explizit vorliegen, könne sie alternativ auch zusammen mit dem Kunden und den Experten aus den jeweiligen Disziplinen formuliert werden. Bei der Formulierung der User Stories werden nicht die konkreten Anforderungen der jeweiligen Experten für Sozialverträglichkeit in den Vordergrund gerückt, sondern die User Stories passend zu allen Anforderungen umformuliert.

Wichtig ist, dass der aus den User Stories entstandene Funktionsteil später mit klar formulierten Kriterien getestet werden kann, um dessen Funktion und Erfüllung im Rahmen des Programmes nachzuvollziehen. Deshalb empfiehlt es sich, den Test gleich mit der User Story festzulegen. In Ergänzung zum herkömmlichen XP werden nicht nur funktionelle Tests wie der Funktionstest und der Unit Test festgelegt, sondern auch ein Test zur Überprüfung der sozialverträglichen Gestaltung integriert. Dieser Test trägt den Namen Kassler-Methodik-Test. Die Kriterien zur sozialverträglichen Gestaltung der jeweiligen User Stories werden bei deren Erstellung festgehalten.

Als Nächstes wird den User Stories eine Priorität zugeordnet. Dabei spielen sowohl die abgeschätzte Entwicklungszeit wie auch die Notwendigkeit der einzelnen User Stories für den gesamten Code eine wichtige Rolle. Gemäß dieser Priorität kann die Reihenfolge der zu implementierenden User Stories festgelegt werden. Die Experten für Sozialverträglichkeit haben hierauf keinen direkten Einfluss. Bei der Festsetzung der Prioritäten arbeitet der Kunde intensiv mit den Entwicklern zusammen. Aus den Prioritäten und den benötigten Entwicklungszeiten kann der Kunde dann einen Release Plan festlegen.

5.2.2 Aktivität: Prototyp ausleiten

In der Protoyping-Phase wird die Software entwickelt. Dabei werden die einzelnen User Stories eventuell noch weiter aufgeteilt und in einzelnen Units implementiert. Mit dem Begriff Unit beziehen wir uns auf die Definition aus der ANSI/IEEE Std 1008-1987:

“A set of one or more computer program modules together with associated control data, (for example, tables), usage procedures, and operating procedures that satisfy the following conditions:

1. All modules are from a single computer program
2. At least one of the new or changed modules in the set has not completed the unit test
3. The set of modules together with its associated data and procedures are the sole object of a testing Process“

Zu jeder Unit wird auch immer ein Unit-Test entwickelt, um jederzeit die Funktion der einzelnen Units überprüfen zu können. Da bei XP der Code häufig umgestaltet wird und ein dynamischer Wechsel der Programmierer möglich sein soll, ist ein solcher Test zwingend notwendig und er wird unabhängig von dem der Expertenvalidierung, eventuell täglich, ausgeführt. In Rahmen der VENUS-Entwicklungsmethode ist es außerdem notwendig zu jeder Unit die ihr übergeordneten User Story sowie deren Testkriterien verfügbar zu haben. Nach dem Erstellen eines Prototyps wird überprüft ob eine Unit noch den Anforderungen zur sozialverträglichen Gestaltung entspricht.

Zur Durchführung von Tests an der Software ist eine aktuelle Version der Software nötig. Im Rahmen von XP werden Tagesversionen regelmäßig erstellt. Zur Durchführung des Tests wird die Software ausgeführt und die Funktion gemäß den User Stories überprüft. In dieser Phase können auch Abweichungen zwischen den eigentlichen (und umformulierten) Kundenwünschen und den festgehaltenen User Stories offenbar werden, wenn der Kunde in der erstellten Version seine Vorstellungen nicht zufriedenstellend umgesetzt sieht. Außerdem können diese aktuellen Versionen genutzt werden, um in der Expertenvalidierung die soziotechnischen Anforderungen zu überprüfen.

5.2.3 Aktivität: Evaluationsergebnisse einarbeiten

In der Testauswertung werden die Abweichungen vom erwarteten Verhalten festgehalten. Da XP ein iterativer Prozess ist, wird mehr als ein Zyklus durchgeführt, was Anpassungen an der Software ermöglicht, bis sie den Kundenwünschen und den Anforderungen zur sozialverträglichen Gestaltung entspricht. Die Abweichungen von den Kundenwünschen, ob explizit in User Stories formuliert oder erst nachträglich aufgefallen, führen zu überarbeiteten User Stories. Diese werden nach jeder Anpassung erneut von den Experten aus den Disziplinen zur sozialverträglichen Gestaltung überprüft. Anschließend wird der gesamte Prozess von XP erneut durchgeführt, einschließlich dem neuen zuordnen von Prioritäten gemäß dem jeweiligen Risiko. Der Prozess wird so lange wiederholt, bis das Ergebnis des Testabschusses eine (hinreichende) Übereinstimmung mit den Kundenwünschen ergibt.

5.3 VB: Konnektor Scrum

Produkt	Applikation, Prototypen
Verantwortliche Rolle	System Engineer
Mitwirkende Rollen	

Da sowohl Scrum als auch die VENUS-Entwicklungsmethode inkrementell und iterativ sind, lassen sich beide Systematiken besonders leicht mit einander verwenden. In kleinen, hierarchisch flachen Teams werden in kurzen Etappen, sogenannten Sprints, kleine Ziele umgesetzt: am Ende einer Iteration, die aus einer kleinen Anzahl an Sprints besteht, steht immer ein lauffähiges System. Dabei wird großer Wert auf regelmäßige Treffen gelegt (Daily Scrum Meetings), in denen der Fortschritt und Probleme in der Entwicklung jedes einzelnen SW-Entwicklers besprochen werden. Der dadurch erreichte große Transparenzgrad und die Möglichkeit, früh auf unerwartete Entwicklungsprobleme oder Fehlplanungen zu reagieren, geben der Methode eine erhöhte Zielorientierung und Flexibilität. Der Kontakt zum Kunden wird durch den sog. Product Owner realisiert, der sowohl die Produktvision kommunizieren muss und eine klare Priorisierung der Anforderungen an das Scrumteam zu vermitteln hat. Er ist es auch, der die regelmäßig aus der Entwicklung ausgeleiteten Ergebnisse eine Sprints bewertet und ggf. neue Richtungen aufzeigt.

Wir werden im Folgenden die Rollen der VENUS-Entwicklungsmethode denen aus Scrum zuordnen. Dabei betrachten wir die Experten als Product Owner, die alle ihre eigenen Ansprüche an das Endprodukt haben, welche durch den Anforderungsanalytiker harmonisiert und ggf. (wie oben beschrieben) priorisiert werden. Der Konnektor Scrum dient dem Zweck, aus der den Produkten der VENUS-Entwicklungsmethode den Scrum-Product-Backlog und User Stories zu erstellen, die Scrum-Rollen den Mitarbeitern zuzuweisen und allgemeine Abläufe von Scrum zu planen und zu präzisieren. Diese Aktivitäten ermöglichen eine Entwicklung mit der Scrum-Methode. Es ist wichtig, dass alle Personen, die später Scrum-Rollen zugewiesen bekommen und damit an der Entwicklung teilnehmen auch hier mitwirken können.

5.3.1 Aktivität: Vorbereitung

Scrum beinhaltet drei Rollen:

- Das Team
- Den ScrumMaster
- Den Product Owner

Das Team besteht aus dem Projektmanager, dem Projektleiter und den SW-Entwicklern. Die Rolle des ScrumMasters übernimmt normalerweise entweder der Projektmanager oder der Projektleiter. Beide sollen allerdings in der Lage sein einander zu ersetzen (wenn der ScrumMaster z. B. krankheitsbedingt oder wegen eines Kundentermins nicht vor Ort sein kann), was schon in der Vorbereitungsphase abgesprochen und geplant wird.

Der ScrumMaster darf nicht die Rolle des Product Owners (und umgekehrt) übernehmen. Im klassischen Scrum ist der Product Owner eine Person, die den Kunden vertritt. Dieser wird auch bei der Einbettung in die VENUS-Entwicklungsmethode benötigt. Allerdings werden ihm die Experten (Anforderungsanalytiker, Trust Engineer, Systemarchitekt, Usability Engineer, Datenschutzverantwortlicher, Informationssicherheitsverantwortlicher) zur Seite stehen und beraten. Stehen vom

Product Owner priorisierte Anforderungen in Konflikt mit denen der Experten, müssen die Konflikte durch die Zusammenarbeit der Experten gelöst und ansonsten eskaliert werden, um eine Entscheidung herbeizuführen. Die Kommunikation zwischen Product Owner und den Experten muss immer dann stattfinden, wenn sich Anforderungen ändern, verworfen werden oder hinzukommen. Es ist daher wichtig die Bedeutung dieser Rolle schon während der Scrum-Vorbereitungs-Phase den Experten zu erläutern. Es ist vom Projekt abhängig ob ein Vertreter des Kunden auch die Rolle des Product Owners übernehmen soll.

Um den Einsatz von Scrum zu ermöglichen, müssen für die erstellten Anforderungen (vgl. Abschnitt 3) User Stories hergeleitet werden. Die User Stories sind eine Grundlage der agilen Entwicklung. Im Gegensatz zum „normalen“ Scrum sollen nicht nur der Kunde, sondern auch die Experten User Stories schreiben bzw. aus den Software Anforderungen ableiten. Es ist wichtig, dass die Experten zusammen mit dem Anforderungsanalysten die User Stories gegenseitig überprüfen und korrigieren bzw. Ungenauigkeiten beseitigen. Es besteht dabei das Risiko, dass die User Stories zu groß sind und den Use Cases zu sehr ähneln. Um das zu vermeiden, sollen alle Beteiligte den Sinn und Zweck der User Stories verstehen und dabei die „INVEST“-Eigenschaften anstreben (Tabelle 8).

I	Independent	User Stories sollen unabhängig voneinander sein
N	Negotiable	User Stories sollen verhandelbar sein
V	Valuable	User Stories sollen einen Wert für den Kunden haben
E	Estimatable	User Stories sollen schätzbar sein
S	Small	User Stories sollen klein sein
T	Testable	User Stories sollen testfähig sein

Tabelle 8: (INVEST-) Eigenschaften der User Stories [109]

Der Projektleiter teilt die User Stories in Tasks (kleinere Aufgaben) auf und stimmt diese Aufteilung mit den Experten ab. Das Product Backlog – als Sammlung aller bekannten User Stories – entsteht somit automatisch. Das erste Sprint Backlog und die Entwürfe der weiteren Backlogs sollen auch schon während Scrum Vorbereitungsphase erstellt werden.

Die Mitarbeiter (inklusive Experten) verständigen sich darauf, den folgenden Scrum-Prinzipien zu folgen:

- Transparenz
- Beobachtung und Anpassung
- Timeboxing
- Abschließen von Dingen
- Ergebnisorientierung

Diese Prinzipien sind die Richtlinien, die „in unterschiedlichen Phasen von Scrum angewendet oder beachtet werden müssen“ [110, p. 28].

Projektmanager und Projektleiter zusammen mit dem Kunden einigen sich, wie lange ein Sprint dauern kann (normalerweise zwischen einer und vier Wochen). Da die VENUS-Entwicklungsmethode aufgrund der verschiedenen beteiligten Disziplinen einen erhöhten Kommunikationsaufwand zur

Folge hat [11], sollten sie sich für einen drei- bis vier-wöchigen Zyklus entscheiden. In den Nachbesprechungen nach dem Ausleiten eines Prototyps, können die Zyklen an die Anforderungen des Teams angepasst werden. Der Projektleiter und der Projektmanager sollen auch entscheiden, wann und wo die folgenden Meetings stattfinden:

- Das Sprint Planning Meeting
- Das Daily Scrum
- Die Sprint-Review
- Die Sprint-Retrospektive

Der Projektmanager und der Projektleiter vereinbaren Termine, wann Experten die Treffen besuchen um entsprechend Rückmeldung für den Stand der Entwicklung zu geben. In der Regel vertritt der Product Owner die Experten, der mit ihnen im Vorfeld zusammen konfliktbehaftete Anforderungen diskutiert und Lösungen entwickelt.

Wie bereits oben erwähnt, sollen die Teammitglieder und Experten den ersten und ggf. noch weitere Backlogs erstellen. Die Experten priorisieren dabei die Aufgaben (und erklären, warum bestimmte Aufgaben wichtiger sind oder zuerst ausgeführt werden müssen). Somit bekommt das Team eine Art von „Referenz“-Backlogs, die für die zukünftige Planung benutzt werden können.

5.3.2 Aktivität: Prototyp ausleiten

Da bei Scrum als Ergebnis einer Iteration, welche zwischen vier bis sechs Sprints beinhaltet, immer ein lauffähiges Produkt entsteht, kann regelmäßig ohne großen Mehraufwand ein Prototyp ausgeleitet werden. Dieser kann durch die Experten, die die Rolle der Product Owner übernommen haben, evaluiert werden.

Die Experten werden zu der letzten Review- bzw. Retrospektive-Meeting eingeladen (abhängig von der Größe der Backlogs können mehrere Meetings für einzelne Experten oder eine Meeting für allen Experten stattfinden). Für jede User Story (oder jeden Task) entscheidet der Projektleiter in Zusammenarbeit mit den Experten, ob diese jeweils für den IT-Sicherheit-, Recht-, Vertrauen- und Usability-Experten relevant ist. Die verantwortlichen Entwickler und ggf. Projektleiter erläutern kurz die User Stories, erzählen wie sie unterschiedliche Probleme gelöst haben und zeigen ggf. die Funktionsweise der Programme auf. Die Experten stellen die Fragen und klären alle Details. Die Experten können auch das erste Feedback geben und die vorgestellten Implementierungen bewerten. Um die Validierung durchzuführen, bekommen die Experten die Backlogs (z. B. in elektronischer Form), und ein Protokoll der Meetings.

5.3.3 Aktivität: Validierungsergebnisse einarbeiten

Die Ergebnisse der Validierung werden nun durch den Anforderungsanalytiker zusammengetragen und konfliktbehaftete Punkte in einem Meeting mit allen beteiligten Experten geklärt. Das bereinigte Ergebnis wird nun wieder dem Projektleiter übergeben, der offene Fragen mit dem Anforderungsanalysten diskutiert. Die Ergebnisse einer Evaluierung werden als Grundlage zur Iterations- oder Sprintplanung verwendet und fließen somit wieder zurück zu den SW-Entwicklern. Der Entwicklungsprozess ist beendet, wenn alle Anforderungen umgesetzt wurden. Dies zu beurteilen obliegt sowohl dem Projektleiter im Falle der Funktionalität und den Experten bzgl. der nichtfunktionalen Anforderungen.

5.4 VB: Expertenvalidierung

Besonderheiten der Expertenvalidierung in der VENUS-Entwicklungsmethode:

- Kontinuierliche Überprüfung der Anforderungserfüllung durch die Experten
- Konsolidierte Empfehlungen der Umsetzung

Produkt	Empfehlungen für die Umsetzung
Verantwortliche Rolle	Projektleiter
Mitwirkende Rollen	Jurist Usability Engineer Trust Engineer IT-Sicherheitsexperte

Zur Sicherstellung einer sozialverträglichen Technikgestaltung müssen in den Prototypen alle zuvor erarbeiteten interdisziplinären Anforderungen berücksichtigt worden sein. Sinn und Zweck der Expertenvalidierung ist deshalb eine entwicklungsbegleitende Qualitätssicherung, die den Stand der Umsetzung der interdisziplinären Anforderungen überprüft und Optimierungspotential identifiziert. Zu beachten ist hierbei, dass oftmals verschiedene Optionen für die Umsetzung einzelner Anforderungen existieren. Die Experten sollten daher unterscheiden, ob eine Anforderung noch überhaupt nicht berücksichtigt wurde, oder ob die Umsetzung noch Mängel aufweist. Dafür validieren die unterschiedlichen Experten den aktuellen Stand der Entwicklung, dokumentieren die erfolgte bzw. nicht erfolgte Umsetzung der Anforderungen und erarbeiten Empfehlungen für Gestaltungsänderungen. Diese Empfehlungen werden interdisziplinär abgestimmt und priorisiert und an das Entwicklungsteam zurückgemeldet.

5.4.1 Aktivität: Validierung durch rechtliche Experten

Eine Nichtbeachtung der rechtlichen Muss- und Soll-Vorgaben, kann unter Umständen eine nicht rechtsgemäße Gestaltung zur Folge haben. Deshalb ist es wichtig auch den Prototyp nochmals von rechtlichen Experten auf die Einhaltung aller Anforderungen hin überprüfen zu lassen. So kann direkt am Prototypen evaluiert werden, ob sich die Erfüllung von rechtlichen Anforderungen positiv auf die Sozialverträglichkeit auswirkt. Auf der Grundlage dieser Ergebnisse können dann im weiteren Prozess Empfehlungen für Gestaltungsänderungen aus juristischer Seite gemacht werden.

Die normativen Vorgaben wurden in einem der vorhergehenden Schritte zu technisch formulierten Anforderungen konkretisiert (vgl. Abschnitt 3.1). Dadurch wird eine direkte Umsetzung dieser Anforderungen in den Prototypen und damit eine sozialverträgliche Gestaltung ermöglicht. Durch diese Umsetzungshilfe wird eine abschließende Bewertung durch den rechtlichen Experten vereinfacht, denn es ist davon auszugehen, dass bei der Einhaltung aller Anforderungen auch eine rechtliche Verträglichkeit automatisch gegeben sein sollte. Trotzdem ist die zwischenzeitliche Begutachtung innerhalb der Expertenvalidierung durch die rechtlichen Experten an diesem Punkt wichtig, um zu überprüfen, ob auch während der Entwicklung alle Anforderungen tatsächlich und richtig umgesetzt wurden. Außerdem gibt diese Begutachtung den rechtlichen Experten die Möglichkeit, die in der Praxis existierenden rechtlichen Probleme direkt am Prototyp zu erkennen. Zuletzt werden für eventuell noch vorhandene Konflikte Verbesserungsvorschläge gemacht.

5.4.2 Aktivität: Validierung durch Usability Experten

Parallel zur Implementierung der Software findet eine Experten-basierte Evaluation der Arbeitsergebnisse hinsichtlich Usability statt. Ziel dieses Vorgehens ist es, eine direkte Rückkopplung von Evaluationsergebnissen in den laufenden Implementierungsprozess zu erreichen. Es wird ein Experten-basiertes Vorgehen empfohlen, weil die Rekrutierung und Beteiligung von echten Nutzern für diesen Prozessschritt als zu aufwändig zu bewerten ist. Untersuchungen im Bereich des Usability Engineering haben gezeigt, dass sich die Mehrzahl aller späteren Nutzungsprobleme durch eine frühzeitige Experten-Evaluation identifizieren und vermeiden lassen [72]. Für eine solche Evaluation müssen Prototypen vorliegen, die mindestens eine bestimmte Teilfunktion des Systems abbilden. Vorteile bieten in diesem Zusammenhang agile Software Engineering wie zum Beispiel SCRUM [110], weil diese in regelmäßigen Abständen lauffähige Software bereitstellen. Das Vorgehen lässt sich aber auch parallel zu anderen Methoden umsetzen (vgl. Abschnitte zu den Konnektoren).

Für diese Form der Evaluation stehen, ergänzend zu den im Prozess erarbeiteten und dokumentierten Anforderungen, Konventionen und Checklisten zur Verfügung, anhand derer die Experten die bereichsspezifischen Themen abarbeiten. Empfohlen wird die Anwendung der Dialogprinzipien aus ISO 9241-10. Die Dialogprinzipien sind unmittelbar auf den Nutzungskontext bezogen und bieten im Vergleich zu konkurrierenden Checklisten wie z. B. den Nielsen-Heuristiken [72], Vorteile für direkt projektbezogene Evaluationsergebnisse [32].

Liste der Dialogprinzipien aus ISO 9241-10:

- Aufgabenangemessenheit: Ein interaktives System ist aufgabenangemessen, wenn es den Nutzer unterstützt, seine Arbeitsaufgabe zu erledigen, d. h., wenn Funktionalität und Dialog auf den charakteristischen Eigenschaften der Arbeitsaufgabe basieren, anstatt auf der zur Aufgabenerledigung eingesetzten Technologie.
- Selbstbeschreibungsfähigkeit: Ein Dialog ist in dem Maße selbstbeschreibungsfähig, in dem für Nutzer zu jeder Zeit offensichtlich ist, in welchem Dialog, an welcher Stelle im Dialog sie sich befinden, welche Handlungen unternommen werden können und wie diese ausgeführt werden können.
- Lernförderlichkeit: Ein Dialog ist lernförderlich, wenn er den Nutzer beim Erlernen der Nutzung des interaktiven Systems unterstützt und anleitet.
- Steuerbarkeit: Ein Dialog ist steuerbar, wenn der Nutzer in der Lage ist, den Dialogablauf zu starten sowie seine Richtung und Geschwindigkeit zu beeinflussen, bis das Ziel erreicht ist.
- Erwartungskonformität: Ein Dialog ist erwartungskonform, wenn er den aus dem Nutzungskontext heraus vorhersehbaren Nutzerbelangen sowie allgemein anerkannten Konventionen entspricht.
- Individualisierbarkeit: Ein Dialog ist individualisierbar, wenn Nutzer die Mensch-Technik-Interaktion und die Darstellung von Informationen ändern können, um diese an ihre individuellen Fähigkeiten und Bedürfnisse anzupassen.
- Fehlertoleranz: Ein Dialog ist fehlertolerant, wenn das beabsichtigte Arbeitsergebnis trotz erkennbar fehlerhafter Eingaben entweder mit keinem oder mit minimalem Korrekturaufwand seitens des Nutzers erreicht werden kann.

5.4.3 Aktivität: Validierung durch Vertrauensexperten

Aufgabe der Expertenvalidierung Vertrauen ist es, das System in seiner aktuellen Version (sämtliche Formen von Prototypen bis hin zu einem fast fertigen Produkt) bzgl. der Erfüllung der gestellten vertrauensbezogenen funktionalen Anforderungen hin zu überprüfen. Hauptergebnis der Expertenvalidierung Vertrauen ist eine Checkliste aus der ersichtlich wird, in welchem Umfang und in welcher Qualität die Anforderungen erfüllt wurden. Sofern Mängel festgestellt werden, werden Änderungswünsche formuliert.

Zuerst muss überprüft werden, ob alle vertrauensbezogenen, funktionalen Anforderungen in der aktuellen Version des Systems berücksichtigt wurden. Hierzu sollte von den Entwicklern eine Übersicht der implementierten Änderungen geliefert werden, die dann überprüft werden kann. Sollte bei dieser Überprüfung herauskommen, dass nicht alle Anforderungen vom aktuellen Design adressiert werden, so wird dies für die Vereinbarung der Gestaltungsänderungen notiert.

Nachdem identifiziert wurde, welche der gestellten Anforderungen in der aktuellen Version des Systems erfüllt wurden, gilt es die Erfüllung der Anforderungen auf ihre Qualität zu überprüfen. Dies ist notwendig, weil Design als kreativer Prozess betrachtet wird, was bedeutet, dass eine Anforderung zumeist durch eine Vielzahl verschiedener Designalternativen adressiert werden kann. Der Trust-Engineer hat daher die Aufgabe, die gewählten Designalternativen auf ihre Eignung hin zu überprüfen und ggf. Änderungswünsche zu formulieren. Das hier beschriebene Vorgehen wurde bereits mehrfach angewendet.

5.4.4 Aktivität: Validierung durch IT-Sicherheitsexperten

Im Rahmen der Sicherheitsanalyse wurden sowohl die vorliegende Dokumentation und Quellcode analysiert und bewertet. Durch die Durchführung der Analyse werden Sicherheitserkenntnisse und Optimierungsmöglichkeiten für den evaluierenden Prototyp gewonnen.

Zunächst werden die Sicherheitsziele identifiziert, anhand deren das System darauf geprüft wird, ob diese erreicht wurden. Diese Prüfung beginnt mit der Analyse des Systems. Es wird untersucht aus welchen Komponenten das System besteht und wie diese zusammenarbeiten, um zu sehen welche Sicherheitsmaßnahmen bereits umgesetzt wurden. Im nächsten Schritt wird untersucht, ob die vorgenommen Sicherheitsmaßnahmen ausreichend sind oder das System Schwachstellen aufweist, die für Angriffe ausgenutzt werden könnten.

5.4.5 Aktivität: Empfehlungen für Gestaltungsänderungen erarbeiten

Die Ergebnisse einer entwicklungsbegleitenden Evaluation von Prototypen durch unterschiedliche Experten für eine sozialverträgliche Gestaltung von UC-Systemen sollen die Programmierer dabei unterstützen, bessere Ergebnisse zu erzielen. Um dies zu ermöglichen ist es wichtig, dass die Evaluationsergebnisse der unterschiedlichen Experten untereinander abgestimmt sind und dem Entwicklerteam konsolidierte und einheitliche Ergebnisse in Form von Empfehlungen für die Gestaltung übergeben werden können.

In einem ersten Arbeitsschritt geht es darum, die Formulierung der Evaluationsergebnisse so anzupassen, dass auch fachfremde Personen ohne Zusatzaufwand zu einem vollständigen Verständnis der Ergebnisse kommen können. Dies ist erstens notwendig, um die nachfolgenden Arbeitsschritte in der Expertengruppe effizient durchführen zu können, und zweitens, um am Ende zu

verständlichen und hilfreichen Empfehlungen für Gestaltungsänderungen zu kommen. Für diesen Zweck gehen die Experten gemeinsam alle Ergebnisse durch und Formulieren diese ggf. entsprechend um.

Nachdem die Experten ein gemeinsames Verständnis von allen Evaluationsergebnissen erreicht und dokumentiert haben geht es nun darum, die Ergebnisse im Gesamtzusammenhang zu konsolidieren. Dafür werden erstens übereinstimmende bzw. sich ergänzende Ergebnisse gesammelt und gemeinsam ein zusammenfassendes Ergebnis formuliert. Zweitens werden sich widersprechende Ergebnisse diskutiert und es wird eine Entscheidung für die beste Lösung des bestehenden Konfliktes getroffen und als konsolidiertes Ergebnis festgehalten, das die konfliktbehafteten Einzelergebnisse ersetzt.

Durch die Überführung der Evaluationsergebnisse in Empfehlungen für Gestaltungsänderungen soll sichergestellt werden, dass für das Entwicklerteam ein konstruktiver Input entsteht. Ein SW-Entwickler wird an dieser Teilaktivität beteiligt, um die Umsetzbarkeit der Empfehlungen zu beurteilen und abzusichern. Er hilft auch bei der projektgerechten Formulierung der Empfehlungen. Konkret wird jedes Evaluationsergebnis aus der zuvor erstellten Liste mit vereinbarten Evaluationsergebnissen der Reihe nach in eine Empfehlung für eine Gestaltungsänderung umgesetzt.

Als weiteren Schritt zu einer konsolidierten Liste mit Empfehlungen für Gestaltungsänderungen müssen die Empfehlungen priorisiert werden. Dies erfolgt auf der Grundlage einer Beurteilung der Wichtigkeit der Empfehlung für eine sozialverträgliche Technikgestaltung und dem abgeschätzten Aufwand der Umsetzung der Empfehlung. Eine hohe Priorität bekommen Empfehlungen mit hoher Wichtigkeit, die mit wenig Aufwand umsetzbar sind.

6 Systemevaluation (Phase 5)

Besonderheiten der Systemevaluation in der VENUS-Entwicklungsmethode:

- Evaluation im realen Nutzungskontext
- Evaluation des gesamten soziotechnischen Zusammenhangs
- Interdisziplinär vereinbarte Änderungsvorschläge als Ergebnis

Die Systemevaluation schließt einen Entwicklungszyklus bestehend aus Bedarfsanalyse, Anforderungsmanagement, Designaktivitäten und Implementierung innerhalb der VENUS-Entwicklungsmethode ab. Erreicht wird eine Bewertung der evaluierten Gestaltungslösung. Anders als bei der Expertenvalidierung (vgl. Abschnitt 5.4) von (teilfunktionalen) Prototypen, werden im Rahmen der Systemevaluation voll funktionsfähige Prototypen im realen Nutzungskontext evaluiert. Evaluiert wird dabei nicht nur die Technik, sondern der gesamte soziotechnische Zusammenhang. Der Hauptzweck der Systemevaluation ist es dabei, Empfehlungen für Gestaltungsänderungen zu erarbeiten, die in einem weiteren Gestaltungszyklus in den Bereichen Bedarfsanalyse, Anforderungsmanagement, Designaktivitäten und Implementierung zu bearbeiten sind. Dieser iterative Gestaltungsgedanke zielt darauf ab, die Evaluationsergebnisse so auszuwerten, dass erstens die möglichen Ansatzpunkte für Verbesserungen erkannt und konkrete Empfehlungen für die Umsetzung gegeben werden. In Abhängigkeit von den Ergebnissen können dies neu identifizierte Bedürfnisse, neue Anforderungen, ein modifiziertes Design oder eine geänderte Umsetzung im Rahmen der Implementierung sein. In jedem Fall erfolgt die Rückmeldung in Form von konstruktiven Vorschlägen für Änderungen. Die Systemevaluation kann auch als Werkzeug zur summativen und abschließenden Bewertung am Ende der Gesamtentwicklung dienen. Dies ist insbesondere dann der Fall, wenn im Rahmen der Systemevaluation keine umsetzbaren Potentiale für weitere Verbesserungen identifiziert werden. In diesem Fall beschließt die Systemevaluation den Gestaltungsprozess. Die Systemevaluation integriert Themen und Methoden aller beteiligten Disziplinen. Dazu gehören ein gemeinsames strukturiertes Vorgehen und die inhaltliche Kooperation bei der Durchführung. Ein Hauptanliegen der Systemevaluation ist es dabei, die Evaluationsergebnisse interdisziplinär so abzustimmen, dass am Ende redundanz- und vor allem widerspruchsfreie Ergebnisse stehen. Dabei sorgt das gemeinsame Vorgehen sowohl für Synergieeffekte und Effizienzsteigerungen, berücksichtigt allerdings auch die Notwendigkeit zur ungestörten disziplinären Arbeit.

Das schrittweise Vorgehen der Systemevaluation ist in Abbildung 16 dargestellt. In der Abbildung wird zwischen Aktivitäten und dem entsprechenden Output, der als Input für die nächste Aktivität dient unterschieden. Die empirische Studie und der Workshop zur Vereinbarung der Ergebnisse werden interdisziplinär durchgeführt. Die Datenanalyse erfolgt getrennt für jede der vier beteiligten Disziplinen. Für die Systemevaluation wurde die IT-Sicherheit als neues Thema aufgenommen.

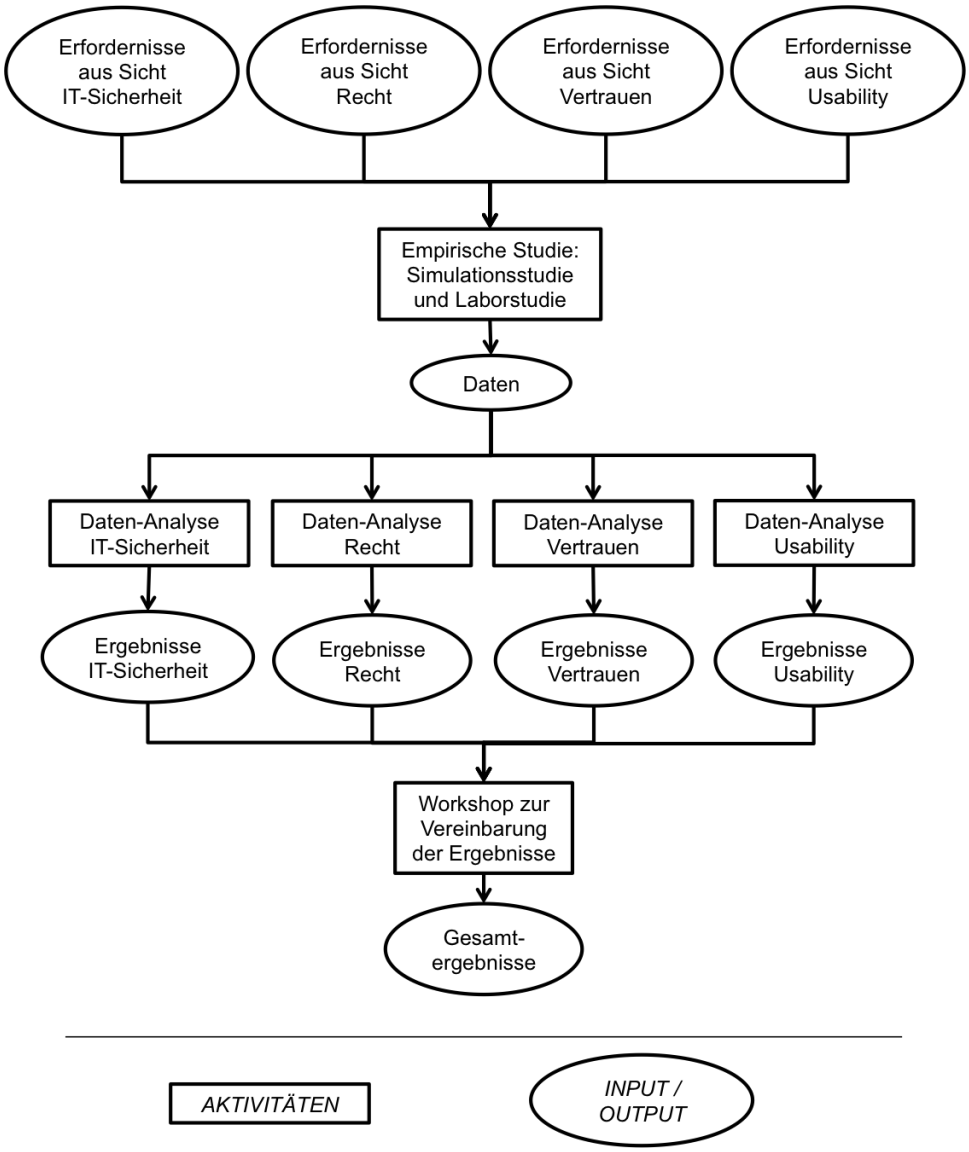


Abbildung 16: Vorgehen bei der Systemevaluation

Die empirische Studie setzt sich aus der Simulationsstudie und der Laborstudie zusammen, die im Folgenden als Vorgehensbausteine beschrieben werden. Im Vorfeld der Durchführung der Systemevaluation ist es notwendig, sich zwischen den Disziplinen über das Setting und die Rahmenbedingungen für die Evaluation abzustimmen. In erster Linie muss das gewählte Setting zu den Anforderungen passen, die sich aus den Eigenschaften des zu evaluierenden Prototyps herleiten. Ebenso wichtig ist es, zu einem Setting zu kommen, das allen beteiligten Disziplinen die für die jeweilige Analyse notwendigen Rahmenbedingungen und im Ergebnis die benötigten Daten und Informationen liefert. Notwendig sind dabei ein realistischer Nutzungskontext und die Integration von echten Nutzern. Weitere Aspekte, wie die Anzahl der beteiligten Nutzer oder die Dauer der Systemevaluation, werden im interdisziplinären Rahmen diskutiert und entschieden. Das resultierende gemeinsame Konzept für die Durchführung der Systemevaluation stellt einen wesentlichen Teil des verfolgten integrativen Ansatzes dar und ermöglicht es, Aufwände zu reduzieren und Synergieeffekte herzustellen.

6.1 VB: Simulationsstudie

Besonderheiten der Simulationsstudie in der VENUS-Entwicklungsmethode:

- Integration von Anwendungsfällen aus verschiedenen Disziplinen wie Recht, Gebrauchstauglichkeit, Vertrauensmanagement
- Aktive Angriffe auf die IT-Sicherheit
- Vorbereitung der Simulationsstudie für ein ubiquitäres System erfordert erhöhten organisatorischen Aufwand

Produkt	Informationen und Daten über soziotechnische Zusammenhänge beim Einsatz der Technik
Verantwortliche Rolle	Projektleiter
Mitwirkende Rollen	Simulationsleitung bestehend aus: Jurist, Trust Engineer, Usability Engineer, IT-Sicherheitsexperte

In der Simulationsstudie soll mit einer Technik Erfahrung gesammelt werden, die es noch nicht gibt, und zum anderen sollen Erfahrungen mit Risiken und Folgen gemacht werden, die zu verhindern sind. Ziel der Simulationsstudie ist es, systematisch Erfahrungen zu sammeln nach dem Prinzip „höchstmögliche Realitätsnähe unter Vermeidung von Schäden“.

6.1.1 Aktivität: Planung der Simulationsstudie

Um systematisch Erfahrungen zu sammeln und später zu anwendernahen Gestaltungsvorschlägen zu kommen, erfolgt die Simulationsstudie in Schritten. Diese sind teilweise mehrmals zu durchlaufen.

Im ersten Schritt müssen die sachverständigen Testpersonen ausgesucht und das Anwendungsfeld bestimmt werden. Diese beiden Voraussetzungen müssen für die Technik und die Fragestellungen charakteristisch sein. In der nächsten Phase findet eine Organisations- und Kommunikationsanalyse statt. Dabei werden die Arbeitsabläufe, die Strukturen und die technischen Unterstützungen untersucht, um zuerst die soziotechnischen Problemstellungen und die Organisationsschwierigkeiten zu analysieren. Außerdem wird ein Vergleich zu den bereits vorhandenen Arbeitsunterstützungen gezogen.

Im nächsten Schritt wird die Simulation intensiv vorbereitet. Dazu wird eine Art „Drehbuch“ geschrieben. Es werden einzelne Rollen festgelegt und beschrieben. Außerdem wird eine Ablaufbeschreibung erstellt, sodass fast nichts dem Zufall überlassen wird. Die sachverständigen Testpersonen sollen unbeeinflusst die Technik testen und Erfahrungen mit ihr sammeln. Die Simulationsleitung kann jedoch einige Rollen beschreiben, die das „Drehbuch“ beeinflussen können, um aussagefähige Ergebnisse zu erhalten. Weiterhin werden verschiedene Fälle erstellt, die die sachverständigen Testpersonen mit der Technik durchspielen sollen. Dabei können die Standardfunktionen der Technik, aber auch kritische Nutzungssituationen erprobt werden. Insbesondere die kritischen Nutzungssituationen sind interessant, um die Einhaltung der datenschutzrechtlichen Regelungen und die Überprüfung der Sicherheit der Technik zu testen. Die Fälle können sich anhand der vorab erarbeiteten Kriterien orientieren. In der VENUS-Entwicklungsmethode werden Anwendungsfälle von allen einbezogen Experten in die Simulationsstudie integriert, um Rückschlüsse auf die Rechtsverträglichkeit, Vertrauenswürdigkeit, Gebrauchstauglichkeit und IT-Sicherheit zu ziehen. Ebenfalls wird

ein Beobachtungskonzept erstellt, damit die Simulationsleitung jeden Fall und seine Auswirkungen beobachten kann.

Anschließend wird die Technik aufgebaut und bereitgestellt. Zu beachten ist, dass dies realitätsnah geschieht. Danach erfolgt eine intensive Erprobung der Technik durch die Simulationsleitung. Dabei können schon Fehler in der Technik, wenn beispielsweise einzelne Funktionen nicht ordnungsgemäß ablaufen, bereinigt werden.

6.1.2 Aktivität: Durchführung der Simulationsstudie

Die zu untersuchende Technik wird von den sachverständigen Testpersonen intensiv erprobt, um anhand der Ergebnisse verbessert werden zu können. Daher sollte ein realistisches Szenario, am besten in einer vertrauten Umgebung der Testpersonen, gewählt werden. Um die Ziele zu erreichen, ist es notwendig:

- mit echten Techniksystemen Erfahrungen zu sammeln, die prototypisch voll funktionsfähig entwickelt sind,
- mit echten Anwendern zu arbeiten, die als „sachverständige Testpersonen“ mit ihren Erfahrungen die Technik besser beurteilen können,
- mit echten Problemen umzugehen, die realen Fällen nachgestellt sind,
- mit echtem Arbeitsmaterial zu arbeiten, das ausschließlich hierfür vorbereitet wurde,
- mit echten Angriffen und Pannen Erfahrungen zu sammeln, wobei die Folgen nur innerhalb der Simulation auftreten und nicht real sind,
- mit echten Testfällen zu arbeiten, die ebenfalls keine realen Schäden verursachen.

Die Technikentwickler sollten während der gesamten Simulation dauerhaft in Bereitschaft sein, um eventuell auftretende Fehler der Technik sofort beheben zu können. Während der Simulation wird in Sicherheitstests, die am laufenden System durchgeführt werden, das System auf seine Robustheit bezüglich der IT-Sicherheit getestet. Hierbei wird auch untersucht, wie ggf. entdeckte Schwachstellen für Angriffe ausgenutzt werden könnten und welche Auswirkungen diese hätten. Die Teilnehmer werden während der Simulationsstudie beobachtet, um Daten für die Auswertung zu sammeln.

6.1.3 Aktivität: Nachbereitung der Simulationsstudie

Die sachverständigen Testpersonen werden jeweils im Anschluss an die simulierten Fälle befragt bzw. um schriftliche Einschätzungen gebeten. Zudem wird am Ende der Simulation mit allen Beteiligten ein Auswertungsgespräch geführt. Die Testpersonen können durch ihr Wissen und ihre Erfahrung Aussagen zu den verschiedenen Funktionen der Technik treffen. Außerdem können sie Stellung nehmen zur Bedienoberfläche, zu organisatorischen Vorgaben oder die Technikeinbindung in die Arbeitsabläufe. Ihre Einschätzungen und ggf. Verbesserungsvorschläge fließen im weiteren Verlauf der Systemevaluation in den Experten-Workshop zur Vereinbarung der Ergebnisse mit ein.

Zur Auswertung der Simulationsstudie müssen die durchgeführten Beobachtungen der Simulationsleitung zusammengetragen, Log-Protokolle begutachtet und Dokumentationen der einzelnen Fälle erstellt werden.

Eine Simulation sollte mindestens zweimal durchgeführt werden, um den sachverständigen Testpersonen die Möglichkeit zu geben, die Verbesserungen zu überprüfen.

6.2 VB: Laborstudie

Besonderheiten der Laborstudie in der VENUS-Entwicklungsmethode:

- Verknüpfung subjektiver und objektiver Messmethoden

Produkt	Informationen und Daten für die individuelle Techniknutzung
Verantwortliche Rolle	Usability Engineer
Mitwirkende Rollen	Trust Engineer

Im Rahmen der Laborstudie wird die Interaktion einzelner Teilnehmer der Systemevaluation mit dem zu evaluierenden Prototyp analysiert. Dabei werden in einem Usability-Labor mehrere Parameter messtechnisch erfasst. Ziel dieses Vorgehens ist die Bewertung einzelner Interaktionselemente hinsichtlich der festzustellenden Reaktionen des Nutzers. Dabei werden die Reaktionen des Nutzers erstens durch eine subjektive Befragung und zweitens durch objektiv gewonnene Messwerte ermittelt.

6.2.1 Aktivität: Objektive Messungen

Zum Einsatz kommt eine software-basierte Mimikerkennung auf der Grundlage von Videodateien, die eine Erfassung der vorherrschenden Emotionen ermöglicht. Gleichzeitig wird die Stärke der vorherrschenden Emotion auf der Grundlage von physiologischen Messungen bestimmt. Es werden Messungen der Herzaktivität und der elektrodermalen Aktivität durchgeführt. Für die Identifikation einzelner kritischer Interaktionselemente werden Verfahren zur Blickbewegungserfassung (Eye-Tracking) eingesetzt. Für starke negative Reaktionen der Nutzer werden über das Eye-Tracking die mit der Reaktion verbundenen Interaktionselemente identifiziert.

6.2.2 Aktivität: Befragung

Ergänzt durch den Abgleich mit den subjektiven Befragungsdaten können nun Rückschlüsse auf die Herausforderungen und Probleme der individuellen Techniknutzung bei der Interaktion mit dem Prototyp gezogen werden und als Informationen und Daten für die weitere Analyse abgelegt werden. Die Befragungen zu den im Rahmen der Laborstudie untersuchten Aufgaben und Interaktions-elemente werden direkt im Anschluss an die Aufgabenerledigung, noch im Labor durchgeführt, um eine direkte In-Beziehung-Setzung der objektiven Messdaten und der subjektiven Befragungsdaten sicherzustellen. Die Laborstudie wird im Hinblick auf Fragen der Gebrauchstauglichkeit durchgeführt. Zudem werden vertrauensunterstützende Komponenten evaluiert.

6.3 VB: Datenauswertung

Besonderheiten der Datenauswertung in der Systemevaluation VENUS-Entwicklungsmethode:

- Gemeinsame Empfehlungen der soziotechnischen Experten

Produkt	Informationen und Daten für individuelle Techniknutzung und soziotechnische Zusammenhänge
Verantwortliche Rolle	Projektleiter
Mitwirkende Rollen	Trust Engineer Usability Engineer Jurist IT-Sicherheitsexperte

In der Datenauswertung werden die Daten durch die Experten ausgewertet und am Ende zu einheitlichen Empfehlungen vereint.

6.3.1 Aktivität: Disziplinäre Auswertung für Recht

Für die Auswertung der Rechtsverträglichkeit des Prototyps wird die Simulationsstudie berücksichtigt. Die Protokolle der Beobachtungen und des Abschlussgespräches sowie die Befragungsergebnisse aus der Simulationsstudie werden strukturiert und anhand der Kriterien, die als Vorstufe der Anforderungserhebung erarbeitet wurden, ausgewertet. Anhand dieser Ergebnisse werden neue Gestaltungsvorschläge generiert.

6.3.2 Aktivität: Disziplinäre Auswertung für Usability

Für die Auswertung der im Rahmen der durchgeführten Systemevaluation erhobenen Daten bezüglich der Gebrauchstauglichkeit des Prototyps wird sowohl die Simulations- als auch die Laborstudie berücksichtigt. Die Beobachtungen und Befragungsergebnisse aus der Simulationsstudie werden strukturiert, bewertet und in eine Liste mit Evaluationsergebnissen überführt. Zudem werden auf der Grundlage der Daten aus der Laborstudie für die Gebrauchstauglichkeit kritische Interaktionselemente identifiziert und beschrieben.

6.3.3 Aktivität: Disziplinäre Auswertung für Vertrauen

Für die vertrauensbezogene Auswertung der im Rahmen der durchgeführten Systemevaluation erhobenen Daten wird auf drei verschiedene Datenquellen zurückgegriffen: Beobachtungen im Rahmen der Simulationsstudie, objektive Daten aus der Laborstudie (Eye-Tracking- und physiologische Daten) sowie auf die quantitativen Daten der Laborstudie, die im Rahmen der Kurzbefragung der Teilnehmer generiert wurden. Die Beobachtungen spielen allerdings eine leicht untergeordnete Rolle, da es stark vom Setting der Simulationsstudie abhängt, wie gut Beobachtungen durchgeführt werden können. Wenn die Simulationsstudie beispielsweise auf mehrere Stockwerke eines Gebäudes inklusive einer Arbeitsgruppe in einem anderen Gebäude verteilt ist, können folglich nur in begrenztem Maße Beobachtungen durchgeführt werden.

Zentrales Ziel der vertrauensbezogenen Auswertung der Daten ist es, die Wirksamkeit der zuvor abgeleiteten vertrauensunterstützenden Komponenten zu evaluieren.

Hierfür helfen zum einen die Eye-Tracking-Daten, die Auskunft darüber geben, ob die vertrauensunterstützenden Komponenten überhaupt vom Nutzer wahrgenommen werden, denn nur dann können sie auch das Vertrauen des Nutzers erhöhen. Des Weiteren helfen die physiologischen Daten, die Rückschlüsse auf die Emotionen der Nutzer im Moment des Wahrnehmens zulassen, die Nützlichkeit der vertrauensunterstützenden Komponenten zu beurteilen. Wenn primär negative Emotionen zu beobachten sind, könnte dies zum Beispiel auf eine unwichtige oder schlecht umgesetzte vertrauensunterstützende Komponente hinweisen.

Diese Daten können dann sehr gut mit den Daten aus der quantitativen Befragung der Teilnehmer trianguliert werden. Bei dieser Befragung wurde sowohl die empfundene Wichtigkeit der einzelnen vertrauensunterstützenden Komponenten als auch die Qualität der Umsetzung im System erfasst. Im Falle des Auftretens negativer Emotionen bei der Wahrnehmung einer vertrauensunterstützenden Komponente kann so, zum Beispiel, nachvollzogen werden, ob die Komponente als unwichtig erachtet oder die Qualität der Umsetzung kritisiert wird.

Auf Basis der Ergebnisse der Auswertung der verschiedenen Datenquellen können dann konkrete Ergebnisse formuliert werden, die als Input mit in den anschließenden Workshop zur Vereinbarung der Ergebnisse mit den Vertretern der anderen Disziplinen dienen.

6.3.4 Aktivität: Disziplinäre Auswertung für IT-Sicherheit

Im Rahmen der Sicherheitsanalyse werden sowohl die erhobenen Daten aus der Simulationsstudie als auch die vorliegende Dokumentation und der Quellcode analysiert und bewertet. Durch die Durchführung der Analyse werden Sicherheitserkenntnisse und Optimierungsmöglichkeiten für den zu evaluierenden Prototyp gewonnen. Angenommen, das System weist Schwachstellen auf, werden im nächsten Schritt Gegenmaßnahmen ausgearbeitet. Mit der Auflistung der möglichen Schutzmaßnahmen wird die Sicherheitsanalyse abgeschlossen.

6.3.5 Aktivität: Workshop zur Vereinbarung der Ergebnisse

In dem Workshop zur Vereinbarung von disziplinären Evaluationsergebnissen ist je ein Vertreter aus jeder beteiligten Fachrichtung vertreten. Ausgangspunkt der Arbeit sind die disziplinären Ergebnisse der Systemevaluation. Diese Ergebnisse werden in Form von kurzen textlichen Beschreibungen für den Workshop vorbereitet.

Im ersten Schritt werden die Resultate der einzelnen Disziplinen zusammengeführt, die disziplinübergreifende Verständlichkeit geprüft und ggf. verbessert und Redundanzen aufgelöst. Zudem werden Kategorien gebildet nach denen die Ergebnisse sortiert werden. Ein Hauptziel dieses Schrittes ist es, ein gemeinsames und gegenseitiges Verständnis aller Ergebnisse sicherzustellen. Zudem werden die Ergebnisse vorstrukturiert. Identifizierte Konflikte zwischen Ergebnissen aus unterschiedlichen Disziplinen werden zu diesem Zeitpunkt noch nicht diskutiert, sondern lediglich festgestellt.

In einem zweiten Schritt werden für die Evaluationsergebnisse, die auf Probleme hinweisen, Vorschläge für Änderungen erarbeitet. Teil dieses Vorganges ist eine intensive Diskussion, die dadurch eine konstruktive Ausrichtung bekommt, dass die Teilnehmer gemeinsam nach Lösungen suchen. Es geht also nicht darum, wer mit den eigenen disziplinären Ergebnissen Recht hat, sondern wie man den zuvor ermittelten Herausforderungen am besten in der Gestaltung begegnet.

Im dritten Schritt werden die Ergebnisse des vorangegangenen Arbeitsschritt aufgegriffen: Die erarbeiteten Vorschläge für Änderungen werden von den Teilnehmern priorisiert. Das Ergebnis ist eine Liste mit Vorschlägen für Verbesserungen des Prototyps.

7 Anhang

7.1 Abbildungsverzeichnis

7.2 Tabellenverzeichnis

7.3 Literaturverzeichnis

[1] A. Bernstein, M. Klein und T. W. Malone, „Programming the Global Brain," *Communications of the ACM 55(5),* pp. 41-43, 2012.

[2] E. L. Trist und K. W. Bamforth, „Some Social and Psychological Consequences of the Longwall Method of Coal-Getting: An Examination of the Psychological Situation and Defences of a Work Group in Relation to the Social Structure and Technological Content of the Work System," *Human Relations 4(3),* pp. 3-38, 1951.

[3] R. P. Bostrom und J. S. Heinen, „MIS Problems and Failures: A Socio-Technical Perspective. Part I: The Causes," *MIS Quarterly 1(3),* pp. 17-32, 1977.

[4] R. P. Bostrom und J. S. Heinen, „MIS Problems and Failures: A Socio-Technical Perspective. Part II: The Application of Socio-Technical Theory," *MIS Quarterly 1(4),* pp. 11-28, 1977.

[5] E. Mumford, „A Socio-Technical Approach to Systems Design," *Requirements Engineering 5(2),* pp. 125-133, 2000.

[6] G. Baxter und I. Sommerville, „Socio-technical Systems: From Design Methods to Systems Engineering," *Interacting with Computers 23(1),* pp. 4-17, 2011.

[7] I. Sommerville, Software Engineering, 9 Edition Hrsg., Boston: Addison-Wesley, 2011.

[8] M. Weiser, „The Computer for the 21st Century," *Scientific American 265 (3),* pp. 94-104, 1991.

[9] M. Weiser und J. Brown, „Designing Calm Technology," *PowerGrid Journa 1(1),* pp. 75-85, 1996.

[10] A. Hoffmann, M. Söllner, A. Fehr, H. Hoffmann und J. M. Leimeister, „Towards an Approach for Developing Socio-Technical Ubiquitous Computing Applications," in *Informatik 2011 - Informatik schafft Communities. Beiträge der 41. Jahrestagung der Gesellschaft für Informatik e.V. (GI)*, Berlin, Deutschland, 2011.

[11] D. Comes, C. Evers, K. Geihs, A. Hoffmann, R. Kniewel, J. Leimeister, S. Niemczyk, A. Roßnagel, L. Schmidt, T. Schulz, M. Söllner und A. Witsch, „Designing Socio-Technical Applications for Ubiquitous Computing - Results from a Multidisciplinary Case Study," in *Distributed Applications and Interoperable Systems (DAIS),*, Stockholm , 2012.

[12] M. Atzmueller, B. E. Macek, A. Hoffmann, M. Kibanov, C. Scholz, M. Söllner und G. Stumme, „Connect-U – Development of Ubiquitous Systems for Enhancing Social Networking," in *Interdisciplinary Design of Socio-Technical Ubiquitous Systems*, (im Erscheinen).

[13] S. Hoberg, L. Schmidt, A. Hoffmann, M. Söllner, J. M. Leimeister, C. Voigtmann, K. David, J. Zirfas und A. Roßnagel, „Socially Acceptable Design of a Ubiquitous System for Monitoring Elderly Family Members," in *42. Jahrestagung der Gesellschaft für Informatik*, Braunschweig , 2012.

[14] M. Knieß, Kreativitätstechniken: Möglichkeiten und Übungen, München: Dt. Taschenbuch-Verl., 2006.

[15] E. Schwarz, I. Krajger und R. Dummer, Innovationskompass für klein- und mittelständische Unternehmen: Neue Ideen finden und entwickeln, Wien: Linde Verlag, 2006.

[16] S. Sato und T. Salvador, „Playacting and focus troupes: theater techniques for creating quick, intense, immersive, and engaging focus group sessions," *Interactions,* pp. 35-41, 1999.

[17] J. M. Leimeister, T. Böhmann und H. Krcmar, „IT-Unterstützung bei der Innovationsentwicklung," in *Handbuch Technologie- und Innovationsmanagement: Strategie - Umsetzung – Controlling*, Wiesbaden, Gabler, 2005, pp. 323-340.

[18] M. Hartmann, U. Bretschneider und J. M. Leimeister, „Patients as innovators - The development of innovative ideas with the Ideenschmiede," in *Wirtschaftsinformatik*, Leipzig, 2013.

[19] M. Niederhuber und P. Bart, Systematisches Vorgehen beim Problemlösen - Methoden und Techniken, Geographic Information Technology Training Alliance, 2010.

[20] H. Backerra, Die sieben Kreativitätswerkzeuge K7: Kreative Prozesse anstoßen, Innovationen fördern, Hanser Fachbuch, 1997.

[21] A. Salovaara und P. Mannonen, „Use of Future-Oriented Information in User-Centered Product Concept Ideation," in *INTERACT*, 2005.

[22] A. Hoffmann, H. Hoffmann und J. M. Leimeister, „Nutzerintegration in die Anforderungserhebung für Ubiquitous Computing Systeme," in *Workshop über Selbstorganisierende, adaptive, kontextsensitive verteilte Systeme (SAKS 2010)*, Kassel, 2010.

[23] P. Kipp und U. Bretschneider, „Collaboration Engineering to Improve the Idea Quality in Ideas Communities," in *20th European Conference on Information Systems (ECIS)*, Barcelona, 2012.

[24] S. Zogaj und U. Bretschneider, „ustomer Integration in New Product Development: A Literature Review Concerning the Appropriateness of Different Customer Integration Methods to Attain Customer Knowledge," in *Proceedings of the 20th European Conference on Information Systems (ECIS)*, Barcelona, Spain, 2012.

[25] R. O. Briggs und G.-J. De Vreede, ThinkLets - building blocks for concerted collaboration, Omaha, Neb.: Univ. of Nebraska, Center for Collaboration Science, 2009.

[26] B. W. Boehm, P. Grünbacker und R. O. Briggs, „Developing Groupware for Requirements Negotiation: Lessons Learned," *IEEE Software,* Bd. 18, Nr. 3, pp. 46-55, 2001.

[27] R. Grohowski, C. McGoff, D. Vogel, B. Martz und J. F. Nunamaker, „Implementing Electronic Meeting Systems at IBM: Lessonst Learned and Success Factors," *MIS Quarterly,* Bd. 14, Nr. 4, pp. 327-345, 1990.

[28] I. Zigurs und B. K. Buckland, „A Theory of Task/Technology Fit and Group Support Systems Effectiveness," *MIS Quarterly,* pp. 313-334, 1998.

[29] ISO, *DIN EN ISO 9241-210,* 2010.

[30] ISO, *ISO/TR 16982,* 2002.

[31] R. Kniewel und L. Schmidt, „Das Design ubiquitärer Systeme am Beispiel von MyGroup," in *Reflexionen und Visionen der Mensch-Maschine-Interaktion - Aus der Vergangenheit lernen, Zukunft gestalten: 9. Berliner Werkstatt Mensch-Maschine-Systeme*, Berlin, 2011.

[32] Deutsche Akkreditierungsstelle, „Leitfaden Usability V1.3," 2010. [Online]. Available: http://www.dakks.de/sites/default/files/71-SD-2-007_Leitfaden%20Usability%201.3.pdf. [Zugriff am 18 12 2012].

[33] C. Thomas und N. Bevan, „Usability context analysis: a practical guide," 1996.

[34] K. Behrenbruch, M. Atzmüller, R. Kniewel, S. Hoberg und G. &. S. L. Stumme, „Gestaltung technisch-sozialer Vernetzung in der Arbeitsorganisation: Untersuchung zur Nutzerakzeptanz von RFID-Technologie," in *Gesellschaft für Arbeitswissenschaft e. V. (Hrsg.): Mensch, Technik, Organisation - Vernetzung im Produktentstehungs- und -herstellungsprozess: 57. Kongress der Gesellschaft für Arbeitswissenschaft*, Chemnitz , 2011.

[35] B. A. Nardi, Context and consciousness: activity theory and human-computer interaction, Cambridge, MA, USA: Massachusetts Institute of Technology, 1995.

[36] J. M. Leimeister, Dienstleistungsengineering Und -Management, Berlin: Springer, 2012.

[37] K. Pohl, Requirements Engineering, Heidelberg: Dpunkt, 2008.

[38] A. Dardenne, A. Lamsweerde und S. Fickas, „Goal-directed requirements acquisition," *Sci. Comput. Program.,* Bd. 20, Nr. 1-2, pp. 3-50, 1993.

[39] A. Cooper, The Inmates Are Running the Asylum, Sams Publishing, 2004.

[40] K. Behrenbruch, M. Atzmüller, C. Evers, L. Schmidt, G. Stumme und K. Geihs, „A Personality Based Design Approach Using Subgroup Discovery," in *4th International Conference on Human-Centered Software Engineering*, Toulouse, 2012.

[41] M. Richter und M. D. Flückiger, Usability Engineering kompakt: Benutzbare Software gezielt entwickeln, 2 Hrsg., Spektrum Akademischer Verlag, 2010, pp. 32-34.

[42] ISO, *DIN EN ISO 9241-12,* 1998.

[43] P. Stähler, Geschäftsmodelle in der digitalen Ökonomie, Köln: Eul Verlag, 2001.

[44] A. Roßnagel, Rechtswissenschaftliche Technikfolgenforschung: Umrisse einer Forschungsdisziplin, Baden-Baden: Nomos Verl.-Ges., 1993.

[45] T. Böhmann und H. Krcmar, „Hybride Produkte: Merkmale und Herausforderungen," in *Wertschöpfungsprozesse bei Dienstleistungen: Forum Dienstleistungsmanagement*, Berlin, Gabler, 2007, pp. 240-255.

[46] B. W. Wirtz, Electronic Business, Wiesbaden: Gabler, 2001, p. 211.

[47] J. Nielsen, Usability Engineering, Boston: Academic Press, 1993.

[48] V. Hammer, U. Pordesch und A. Roßnagel, „KORA – Eine Methode zur Konkretisierung rechtlicher Anforderungen zu technischen Gestaltungsvorschlägen für Informations-und Kommunikationssysteme," *Infotech/I+ G,* pp. 21-24, 1993.

[49] V. Hammer, U. Pordesch und A. Roßnagel, Betriebliche Telefon- und ISDN-Anlagen rechtsgemäß gestaltet, Berlin: Springer-Verlag, 1993.

[50] A. Roßnagel, „Rechtswissenschaftliche Gestaltung der Informationstechnik," in *Wissen, Vernetzung, Virtualisierung – Liber Amicorum zum 65. Geburtstag von Univ.-Prof. Dr. Udo Winand*, Köln, Josef Eul Verlag, 2008, pp. 381-390.

[51] K. Larenz, Methodenlehre der Rechtswissenschaft, Berlin: Springer-Verlag, 1991.

[52] Bundesverfassungsgericht, „BVerfGE 65, 1 – Volkszählungsurteil," in *Entscheidungen des Bundesverfassungsgerichts – Band 65*, Karlsruhe, Mohr Siebeck, 1983, pp. 1-71.

[53] Bundesverfassungsgericht, „BVerfGE 120, 274 – Online-Durchsuchung," in *Entscheidungen des Bundesverfassungsgerichts – Band 120*, Karlsruhe, Mohr Siebeck, 2008, pp. 274-350.

[54] M. Söllner, A. Hoffmann, H. Hoffmann und J. M. Leimeister, „Vertrauensunterstützung Für Sozio-Technische Ubiquitäre Systeme," *Zeitschrift für Betriebswirtschaft Special Issue S4,* pp. 109-140, 2012.

[55] A. Abdul-Rahman und S. Hailes, „Supporting Trust in Virtual Communities," in *33rd Annual Hawaii International Conference on System Sciences (HICSS)*, Hawaii, 2000.

[56] B. M. Muir, „Trust in Automation: Part I. Theoretical Issues in the Study of Trust and Human Intervention in Automated Systems," *Ergonomics 37(11),* pp. 1905-1922, 1994.

[57] J. D. Lee und K. A. See, „Trust in Automation: Designing for Appropriate Reliance," *Human Factors 46(1),* pp. 50-80, 2004.

[58] N. Luhmann, Trust and Power, John Wiley and Sons Ltd., 1979.

[59] G. Kotonya und I. Sommerville, „Requirements Engineering with Viewpoints," *Software Engineering Journal 11(1),* pp. 5-18, 1996.

[60] J. Bortz und N. Döring, Forschungsmethoden Und Evaluation, Heidelberg: Springer, 2006.

[61] R. C. Mayer, J. H. Davis und F. D. Schoorman, „An Integrative Model of Organizational Trust," *Academy of Management Review 20(3),* pp. 709-734, 1995.

[62] T. Ebert, „Facets of trust in relationships – a literature synthesis of highly ranked trust articles," *J Bus Mark Manage,* Bd. 3, Nr. 1, pp. 65-84, 2009.

[63] M. Söllner, A. Hoffmann, H. Hoffmann, A. Wacker und J. M. Leimeister, „Understanding the Formation of Trust in IT Artifacts," in *Proceedings of the International Conference on Information Systems (ICIS)*, Orlando Florida, USA, 2012.

[64] B. M. Muir und N. Moray, „Trust in automation. Part II. Experimental studies of trust and human intervention in a process control simulation," *Ergonomics,* Bd. 39, Nr. 3, pp. 429-460, 1996.

[65] S. Zuboff, In the age of smart machines: the future of work technology and power, New York: Basic Books, 1988.

[66] V. Shankar, G. Urban und F. Sultan, „Online trust: a stakeholder perspective, concepts, implications, and future directions," *J Strateg Inf Syst,* Bd. 11, Nr. 3-4, pp. 325-344, 2002.

[67] I. Sommerville, Software Engineering, Harlow: Addison-Wesley, 2007.

[68] L. Chung, B. A. Nixon, E. Yu und J. Mylopoulos, Non-Functional Requirements in Software Engineering, Kluwer, Boston: Kluwer Academic Publishers, 2000.

[69] L. M. Cysneiros, J. C. S. do Prado Leite und J. de Melo Sabat Neto, „A Framework for Integrating Non-Functional Requirements into Conceptual Models," *Requirements Engineering 6(2),* pp. 97-115, 2001.

[70] D. Gross und E. Yu, „From Non-Functional Requirements to Design through Patterns," *Requirements Engineering 6(1),* pp. 18-36, 2001.

[71] J. Cleland-Huang, R. Settimi, O. BenKhadra, E. Berezhanskaya und S. Christina, „Goal-Centric Traceability for Managing Non-Functional Requirements," in *27th International Conference on Software Engineering (ICSE)*, St. Louis, 2005.

[72] J. Nielsen, „Heuristic Evaluation," in *Usability Inspection Methods*, New York, John Wiley & Sons, 1994, pp. 25-62.

[73] K. Behrenbruch, S. Jandt, L. Schmidt und A. Roßnagel, „ Normative Anforderungsanalyse für ein RFID-basiertes Assistenzsystem für Arbeitsgruppen.," in *Gesellschaft für Arbeitswissenschaft e. V. (Hrsg.): Gestaltung nachhaltiger Arbeitssysteme - Wege zur gesunden, effizienten und sicheren Arbeit: 58. Kongress der Gesellschaft für Arbeitswissenschaft*, Dortmund, 2012.

[74] R. O. Briggs und P. Gruenbacher, „EasyWinWin: Managing Complexity in Requirements Negotiation with GSS," *Proceedings of the 35th Hawaii International Conference on System Sciences,* 2002.

[75] R. Likert, „A Technique for the Measurement of Attitudes," *Archives of Psychology,* Bd. 140, p. 1–55, 1932.

[76] S. Robertson und J. Robertson, Mastering the Requirements Process, Boston: Addison-Wesley, 2006.

[77] X. Franch, C. Palomares, C. Quer, S. Renault und F. De Lazzer, „A Metamodel for Software Requirement Patterns," in *16th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ)*, Essen, Germany, 2010.

[78] L. Chung, B. Paech, L. Zhao, L. Liu und S. Supakkul, „Repa Requirements Pattern Template V1.0.1.," in *Second International Workshop on Requirement Pattern (RePa)*, Chicago , 2012.

[79] A. Hoffmann, M. Söllner, H. Hoffmann und J. M. Leimeister, „Requirement Patterns to Support Sociotechnical System Design," in *Interdisciplinary Design of Socio-Technical Ubiquitous Systems*, Springer, (im Erscheinen).

[80] S. Renault, O. Mendez-Bonilla, X. Franch und C. Quer, „A Pattern-Based Method for Building Requirements Documents in Call-for-Tender Processes," *International Journal of Computer Science and Applications 6(5),* pp. 175-202, 2009.

[81] A. Hoffmann, T. Schulz, J. Zirfas, H. Hoffmann, A. Roßnagel und J. M. Leimeister, „Rechtsverträglichkeit Als Eigenschaft Soziotechnischer Systeme – Ziele Und Anforderungsmuster," *Wirtschaftsinformatik | Business and Information Systems Engineering,* im Review.

[82] A. Hoffmann, H. Hoffmann und M. Söllner, „Fostering Initial Trust in Applications - Developing and Evaluating Requirement Patterns for Application Websites," in *21th European Conference on Information Systems (ECIS)*, 2013.

[83] A. Hoffmann, M. Söllner, H. Hoffmann und J. M. Leimeister, „Deriving Requirement Patterns to Increase User's Trust in Sociotechnical Systems," *Transaction on Management Information Systems,* im Review.

[84] C. Palomares, C. Quer, X. Franch, C. Guerlain und S. Renault, „A Catalogue of Non-Technical Requirement Patterns," in *IEEE Second International Workshop on Requirements Patterns (RePa)*, Chicago , 2012.

[85] ISO, *DIN EN ISO 9241-210,* Berlin: Beuth, 2006.

[86] J. J. Garrett, Die Elemente der User Experience, New York, NY: AIGA, 2008.

[87] B. Baxley, Universal model of a user interface, 2003.

[88] M. Koch, H. Reiterer und A. Tjoa, Software-Ergonomie: Gestaltung von EDV-Systemen - Kriterien, Methoden und Werkzeuge, Wie-New York: Springer, 1991.

[89] R. Kniewel und C. Evers, Challenging the need for Transparency, Controllability, and Consistency in Usable Adaption Design, 2013.

[90] H. Gomaa, Designing Concurent, Distributed, and Real-Time Applications with UML, Addison-Wesley, 2004.

[91] A. Cooper, D. Cronin and R. Reimann, About Face - The essentials of interaction design, Indianapolis, IN: Wiley Publ Inc., 2007.

[92] ISO, *DIN 6600,* 1966.

[93] T. Stapelkamp, Screen- und Interfacedesign. Gestaltung und Usability für Hard- und Software, Berlin: Springer, 2007.

[94] B. &. P. C. Shneiderman, Designing the user interface: Strategies for effective, Reading, MA: Addison-Wesley, 2009.

[95] J. Tidwell, Designing interfaces, Beijing ; Sebastopol, CA: O'Reilly, 2006.

[96] C.-E. Dessart, V. Genaro Motti und J. Vanderdonckt, „Showing user interface adaptivity by animated transitions," in *ACM (ed.) EICS '11 Proceedings of the 3rd ACM SIGCHI symposium on Engineering interactive computing systems*, Pisa, Italy, 2011.

[97] K. Höök, „Steps to take before intelligent user interfaces become real,“ *Interacting with computers 12(4),* pp. 409-426, 2000.

[98] T. Lavie und J. Meyer, „Benefits and costs of adaptive user interfaces,“ *International Journal of Hu-man-Computer Studies 68(8),* pp. 508-524, 2010.

[99] M. Peissner und T. Sellner, „Transparency and controllability in user interfaces that adapt during run-time,“ in *CHI 2012 Workshop: End-user Interactions with Intelligent and Autonomous Systems*, Austin, 2012.

[100] R. Kniewel, C. Evers, L. Schmidt und K. Geihs, „Designing Usable Adaptations,“ in *Interdisciplinary Design of Socio-Technical Ubiquitous Systems*, Springer, (im Erscheinen).

[101] D. S. McCrickard und C. M. Chewar, „Attuning notification design to user goals and attention costs,“ *Commun. ACM 46(3),* pp. 67-72, 2003.

[102] D. M. Brown, Communicating design. Developing web site documentation for design and planning, Berkeley, CA: Peachpit Press , 2007.

[103] S. Hoober und E. Berkman, Designing Mobile Interfaces, 2011.

[104] H. Baraki, K. Geihs, A. Hoffmann, C. Voigtmann, B.-E. Macek und J. Zirfas, „Towards Interdisciplinary Design Patterns for Ubiquitous Computing Applications,“ ITeG, Universität Kassel.

[105] T. Stahl, M. Völter, S. Efftinge und A. Haase, Modellgetriebene Softwareentwicklung, Heidelberg: dpunkt.verlag GmbH, 2007.

[106] J. Bettin, „MDSD-Prozessbausteine und Best Practices,“ in *Modellgetriebene Softwareentwicklung*, Heidelberg, dpunkt.verlag GmbH, 2007, pp. 215-245.

[107] C. Bunse und A. von Knethen, Vorgehensmodelle Kompakt, Spectrum Akademischer Verlag, 2002.

[108] K. Beck, eXtreme Programming Explained (2.Auflage), Addison-Wesley, 2004.

[109] M. Cohn, User Stories Applied For Agile Software Development, Addison-Wesley, 2004.

[110] R. Wirdemann, SCRUM mit User Stories, München Wien: Carl Ahnser Verlag, 2011.