Bachelorarbeit am Institut für Informatik der Freien Universität Berlin,

Arbeitsgruppe Software Engineering

Benachrichtigungen im semantischen Annotationssystem AnnoSys

Sascha Gennrich

Matrikelnummer: 4301150

s.gennrich@fu-berlin.de

Gutachter: Prof. Dr. Lutz Prechelt

Zweitgutachter: Prof. Dr. Adrian Paschke

Betreuer: Lutz Suhrbier

Berlin, 1. April 2014

Zusammenfassung

Diese Arbeit entstand in Zusammenarbeit mit dem AnnoSys-Projekt am Botanischen Garten und Botanischen Museum Berlin-Dahlem. Mit der AnnoSys-Webanwendung können Nutzer Annotationen für Biodiversitäts-Daten (digitale Herbarbelege) erstellen. Im Rahmen dieser Arbeit wurde ein Benachrichtigungssystem entwickelt, das private Nachrichten in der Webanwendung sowie E-Mails an die Nutzer schickt, um diese über neu erstellte Annotationen anderer Nutzer zu informieren. Durch die Auswahl von Kriterien die auf eine bestimmte Annotation zutreffen sollen, können die Nutzer die Gruppe von Annotationen einschränken für die sie Benachrichtigungen erhalten wollen. Durch den wiederholten Prozess von Implementierung und Nutzertests wurde die Gebrauchstauglichkeit (Usability) dieser Funktion gesichert und verbessert. Es wird erläutert wie Nutzertests durchgeführt werden und wie aus den Ergebnissen dieser Tests Lösungsvorschläge zur Verbesserung gezogen werden.

Eidesstattliche Erklärung

Ich versichere hiermit an Eides Statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

01.04.2014

Sascha Gennrich

Inhaltsverzeichnis

1 Einführung

Herbarbelege oder auch Herbarbögen, sind eine wichtige Grundlage der Biodiversitätsforschung. Sie erlauben es Forschern Pflanzen unterschiedlicher Herkunft zu vergleichen und zu überprüfen[1]. Ein Herbarbeleg besteht aus der getrockneten und gepressten Pflanzenprobe, Informationen über die Pflanze und weiteren Metadaten, die der Sammler während des Sammelns hinzugefügt hat. Zu den häufigen Daten, die auf einem Herbarbeleg zu finden sind, gehören der wissenschaftliche Name der Pflanze, der Name des Sammlers sowie der Fundort (heutzutage mit GPS-Koordinaten). Außerdem ist es üblich, dass ein Sammler jedem seiner Herbarbelege eine eindeutige Sammelnummer zuweist.
Eine Sammlung von Herbarbelegen wird als Herbarium bezeichnet. Das Herbarium des Botanischen Gartens und Botanischen Museums (BGBM) „Herbarium Berolinense“ umfasst ca. 3,6 Millionen Herbarbelege unter anderem von Blütenpflanzen, Algen, Moose und Farne[2].
Ein Teilgebiet der Biologie ist die Taxonomie. Dieses Gebiet beschäftigt sich mit der Klassifizierung von Lebewesen und erfasst ihre verwandtschaftlichen Beziehungen in einer Systematik. Das Bestimmen von Arten stellt selbst für erfahrene Taxonomen eine Herausforderung dar und ist ein bekanntes Problem. Häufig leihen sich Taxonomen, oder auch andere Interessierte, Herbarbelege einer Sammlung aus und stellen fest, dass die aufgeklebte Pflanzenprobe nicht zu der Art gehört, die in den Metadaten angegeben ist oder sich der Name oder die Familie der Art im Laufe der Zeit verändert hat. Daraufhin wird ein Vermerk auf dem Beleg hinterlassen und an die Sammlung, mit Bitte auf Überprüfung durch den Kurator, zurück gesendet.
In vielen Herbarien gibt es Anstrengungen Herbarbelege zu digitalisieren und öffentlich zugänglich zu machen [5]. Diese Digitalisierung ermöglicht es Botanikern, Taxonomen und anderen Forschern auf der ganzen Welt die Proben ohne großen Aufwand einzusehen und zu überprüfen. Derzeit gibt es kein standardisiertes Verfahren, um Änderungen an den Metadaten vorzuschlagen [14]. Um Änderungen am Beleg vorzuschlagen, beschränken sich viele Portale auf E-Mailkommunikation oder Webseitenkommentare. Auf der JSTOR Global Plants Webseite können beispielsweise Diskussionen über einen Eintrag via Disqus[3] geführt werden[4]. Für die Kuratoren von solchen Sammlungen stellt es einen großen Aufwand dar, diese Kommentare zu prüfen, die nötigen Informationen herauszufiltern und ggf. die Daten des Herbarbelegs zu aktualisieren.
Das von der Deutschen Forschungsgemeinschaft geförderte Projekt AnnoSys[5] am BGBM entwickelt eine Webanwendung, um solche Kommentare in strukturierte Form zu erstellen und zu veröffentlichen.
Im Rahmen dieser Arbeit wird ein Modul entwickelt, welches Nutzern und Kuratoren

[1] 100 Uses For A Herbarium – `http://www.virtualherbarium.org/vh/100UsesASPT.html`, Stand: 28.03.2014
[2] http://bgbm.org/de/wissenschaft21-herbarium
[3] Disqus ist ein webseitenübergreifender Kommentarservice. Nutzer können sich einmal anmelden und anschließend auf jeder Webseite, welche Disqus einbindet, Kommentare schreiben
[4] Beispieleintrag für eine Diskussion bei JSTOR - http://plants.jstor.org/ specimen/ea000001244
[5] AnnoSys - `http://annosys.bgbm.org`

erlaubt sich für eine Gruppe von Datensätze anzumelden, für die sie über neu veröffentlichte Kommentare informiert werden wollen. Um die Nutzbarkeit des Moduls zu gewährleisten, wird ein iterativer Entwicklungsprozess gewählt, wobei regelmäßige Nutzertests im Fokus stehen.
In Kapitel 2 werden zuerst die Anforderungen an das Modul und dann die verwendeten Komponenten sowie das AnnoSys-System beschrieben. Außerdem werden die Grundlagen für Usability-Tests dargelegt und wichtige Punkte, die bei ihrer Durchführung beachtet werden sollten, hervorgehoben. In Kapitel 3 wird der iterative Entwicklungsprozess beschrieben. Dieser besteht aus Prototypentwicklung, Usabiltiy-Tests und Verbesserung des Prototyps auf Basis der Ergebnisse dieser Tests, die während der Entwicklung statt fand. Zum Schluss werden in Kapitel 4 die Ergebnisse des Entwicklungsprozesses, aus dem vorigen Kapitel zusammengefasst sowie eine Bewertung der verwendeten Technologien abgegeben.

2 Grundlagen

2.1 Anforderungen

Die Anforderungen an das Benachrichtigungssystem wurden aufgestellt indem die AnnoSys-Entwickler befragt und vorhandene technische Spezifikationen berücksichtigt wurden.
Ein Nutzer, ist eine Person, die sich im System registriert hat und mit diesem interagieren möchte. Die Zielgruppe kann in zwei Gruppen eingeteilt werden. Zum einen in die Gruppe der Annotierenden, welche Annotationen für Datensätze erstellen wollen und zum anderen die Kuratoren, welche für Sammlungen von Datensätzen verantwortlich sind. Die Kuratoren stellen sicher, dass Daten einer Annotation dem Herbarbeleg hinzugefügt werden, sofern diese für korrekt befunden wurden.
Das Ziel ist es ein Modul zu entwickeln, das es ermöglicht, Nutzer darüber zu informieren, dass neue Annotationen erstellt wurden und diese in einer verständlichen Art zu präsentieren. Darüber hinaus sollen Kuratoren aufgefordert werden die Annotation in ihre Datenbank zu übernehmen und ggf. auch ein Annotationsetikett auf den physikalischen Beleg ihrer Sammlung zu kleben.
Die Nutzer sollen Annotationen nach bestimmten Kriterien auswählen können für die sie Benachrichtigungen erhalten wollen. Kriterien sollen unter anderem für den wissenschaftlichen Namen der Art, die Familie, den Sammler oder die Sammlung gemacht werden können. Wird dann eine Annotation erstellt, auf die diese Kriterien zutreffen, so sollen die Nutzer benachrichtigt werden. Diese Auswahl von Kriterien wird im Rahmen dieser Arbeit als Abonnement bezeichnet.
Die Benachrichtigungen sollen per E-Mail versendet werden sowie in der Web-Applikation erscheinen. Außerdem sollen sich Nutzer gegenseitig Nachrichten schicken können, um bei Unstimmigkeiten oder Unklarheiten im Bezug auf den Inhalt der Annotation diese klären zu können. In Tabelle 1 sind alle Anforderungen aufgelistet.

2.2 AnnoSys

AnnoSys ist eine Webanwendung zum Annotieren von Biodiversitätsdaten. Die Oberfläche wird mit dem Eclipse Remote Application Framework (RAP) realisiert. RAP implementiert die Java-GUI-Schnittstellen Software Widget Toolkit (SWT) und JFace zur Darstellung im Browser und erlaubt so, mit wenig Aufwand für Java-Entwickler, Weboberflächen zu schreiben.
Die Daten der Annotationen und Annotierenden werden als RDF-Graph in einem Triplestore gespeichert. Die Annotationen werden gemäß der Spezifikation der Open Annotation Community Group[6] erstellt, um eine einheitliche Sprache für den Austausch von Annotation zu gewährleisten. Die digitalen Herbarbelege werden in einer XML-Datenbank gespeichert.
AnnoSys richtet sich an Taxonomen, Systematiker und Wissenschaftler, die in ihrem Arbeitsalltag mit Herbarbelegen arbeiten. Die Anwendung wurde mit Blick auf die Arbeitsabläufe und Bedürfnisse dieser Gruppen entwickelt. Um digitale Herbarbelege in

[6]Open Annotation - `http://www.w3.org/community/openannotation`

Tabelle 1: Anwendungsanforderungen

#	Sicht	Anforderung
1	Nutzer	Benachrichtigungen per Mail erhalten
2		Benachrichtigungen in der Anwendung anzeigen
3		Abonnements, durch Auswahl von Kriterien einrichten können
4		Nachricht an einen Annotierenden schicken
5	Annotierender	Mail über erfolgreiche Erzeugung und Inhalt einer Annotation erhalten
6		Benachrichtigung, wenn ein Kurator eine meiner Annotation bearbeitet hat
7	Kurator	Nachricht erhalten, wenn ein Datensatz einer kuratierten Sammlungsdatenbank annotiert wurde.
8	Technisch	Nachrichten persistent in JMS-Queues speichern
9		Apache Camel/Apache ActiveMQ verwenden, um Nachrichten zu versenden

AnnoSys zu importieren, müssen diese im abcd-Datenformat vorliegen. Das abcd-Schema ist als Austauschformat von Herbardaten, DNA-Proben, aber auch Beobachtungsdaten die keine physikalischen Sammlungsproben produzieren, gedacht [4].
In Abbildung 1 ist ein Screenshot der Webseite zum Annotieren von digitalen Herbarbelegen zu sehen. Wenn im Folgenden von Datensätzen die Rede ist, sind diese digitalen Herbarbelege gemeint. Die Seite ist in drei Spalten unterteilt. (1) ist die Annotationsübersicht. Dort werden Annotationen und Datensätze angezeigt, die aktuell im mittleren Arbeitsbereich (2) geöffnet sind. Auf der rechten Seite befindet sich die Profilübersicht (3). Das ist der private Bereich des Nutzers, in dem Nachrichten, unveröffentlichte Annotationen und Abonnements angezeigt werden.
Ein Nutzer der einen Datensatz annotieren möchte wird von einer externen Webseite zur AnnoSys-Webanwendung weitergeleitet. Er hat z. B. einen Datensatz, auf der Herbariumswebseite des BGBMs[7] angesehen und festgestellt, dass er etwas kommentieren möchte.
Auf der AnnoSys-Seite wird im Arbeitsbereich dann ein entsprechendes Formular geöffnet. Dort wählt der Nutzer als erstes einen Annotationstyp, um sein Kommentar zu kategorisieren. Beispiele für Annotationstypen sind: Bestimmung oder Aufsammlung, um z. B. den Fundort zu kommentieren. Je nach Typ wird eine angepasste Eingabemaske präsentiert, in der die Kommentare strukturiert eingegeben werden können. Im Rahmen der AnnoSys-Anwendung werden diese strukturierten Kommentare Annotation genannt. Über die Schaltfläche „Veröffentlichen“ in der Toolbar wird die Annotation gespeichert und für andere Nutzer öffentlich sichtbar.

[7]Beispiel eines Herbarbelegs im virtuellen Herbar des BGBM. `http://ww2.bgbm.org/Herbarium/specimen.cfm?Barcode=B100295708`

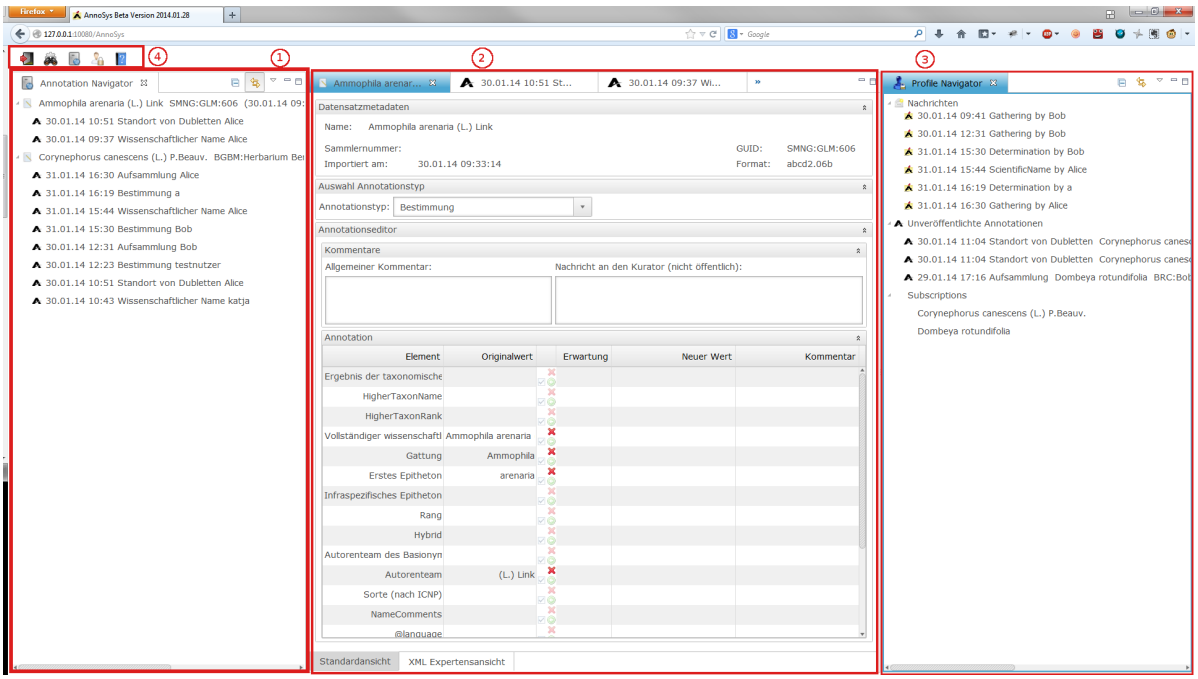


Abbildung 1: Annosys-Webseite: (1) Annotationsübersicht, (2) Arbeitsbereich, (3) Profilübersicht, (4) Toolbar

2.3 Technische Grundlagen

Die groben Anforderungen an das Benachrichtigungssystem wurden bereits im DFG-Projektantrag von 2011 beschrieben. Daraufhin wurde die Entscheidung getroffen Camel als Routing-Engine für die Benachrichtigungen mit ActiveMQ als Nachrichtenspeicher zu verwenden. Auf den ersten Blick scheint es genau das richtige für diesen Anwendungsfall zu sein: Nachrichten an einzelne Nutzer und an Gruppen von Nutzern schicken.
Zum Ende der 90er Jahre entwickelten sich Softwaresysteme von zentrierten Anwendungen hin zu einem Verbund von verteilten, lose gekoppelten Softwarekomponenten die miteinander interagierten. In diesen Systemen wurde die sogenannte Message Oriented Middleware (MOM) immer populärer. MOM beschreibt eine Softwarearchitektur die Mithilfe von Nachrichten Status bzw. Aufgaben zwischen den Komponenten übermittelt. Ein Klient eines MOM-Systems kann Nachrichten an eine beliebige Anzahl anderer Klienten verschicken bzw. von anderen empfangen; synchron wie asynchron. Außerdem beschreibt es ein Anmelde-Versende-Verfahren, bei dem der Sender nicht darüber informiert ist, wer Teil der Empfängergruppe ist. Dies ist ähnlich zum IP-Multicast. [1]

JMS und ActiveMQ Das „Java Message System“ (JMS) bietet eine Java-Schnittstelle um diese Art von Nachrichten basierten Systemen zu realisieren. Es gibt eine Vielzahl von Anbietern die JMS implementieren[8], beispielsweise „Apache ActiveMQ“. ActiveMQ läuft als eigenständiger Prozess und verwaltet die verschiedenen Nachrichten-Queues die

[8] de.wikipedia.org/wiki/Java_Message_Service#JMS-Provider

mit Camel angesprochen werden können.

Apache Camel Eine weitere Komponente ist Apache Camel[9]. Camel ist ein Enterprise Integration Pattern Framework und implementiert alle Pattern aus dem Buch *Enterprise Integration Patterns* [9]. Camel erlaubt mit relativ einfacher Domain Specific Language in z. B. Java, Scala oder Spring Routing zwischen verschiedenen Softwarekomponenten zu definieren. In Camel werden die Komponenten als Endpoints bezeichnet.

Endpoint Ein Endpoint kann beispielsweise eine Datei sein, aus der gelesen wird oder die Adresse eines SMTP-Servers über den Mails versendet werden sollen. In Camel werden Endpoints über URIs identifiziert. Der erste Teil der URI entspricht dem Namen der Komponente bzw. Schemas, der Rest der URI ist abhängig von der Art des Endpoints. Listing 1 zeigt die Einrichtung einer Route für Camel in Java. Hier wird der Text aus einer Datei eingelesen, von einer Prozessorkomponente aufgearbeitet und anschließend in eine Datenbank geschrieben[10].

Listing 1: Camel Routendefinition um Daten aus einer Datei in eine Datenbank zu schreiben

```
from("file:myfile.data").process(new MyDatabaseProcessor()).
    to("jdbc:sqldatabase")
```

Anstelle von Dateien und Datenbankverbindungen, können mit ActiveMQ Queues und Topics verarbeitet werden.

Queues ActiveMQ-Endpoints setzen sich aus drei Komponenten zusammen: (1) Schema-Prefix `activemq`, (2) Angabe, ob es eine Queue oder ein Topic sein soll und (3) dem Namen der Queue oder des Topics. Queues verarbeiten Nachrichten nach dem „First in-First Out“-Prinzip. Wird eine Nachricht an einen Queue-Endpoint gesendet, wird diese gemäß der Definition der Route verarbeitet. Sind mehrere Routendefinitionen vorhanden, so wird nur eine ausgewählt, die die Nachricht bearbeitet [12, S. 26]. Im folgenden Listing wird eine Nachricht, die an die Queue namens „test“ gesendet wird, als Datei im Ordner „output“ abgespeichert.

```
from("activemq:queue:test").to("file:output")
```

Topics Im Gegensatz zu Queues entsprechen Topics dem Anmelde-Versende-Prinzip. Treffen mehrere Routendefinitionen auf einen Endpoint zu wird die Nachricht vervielfacht und von allen bearbeitet.

```
from("activemq:topic:test").to("file:output");
```

[9] Apache Camel - `http://camel.apache.org`

[10] Zur Übersicht wurden Codeteile weggelassen die Camel und ActiveMQ initialisieren. Ein komplettes Beispiel findet sich im Anhang auf Seite 39.

Camel bietet noch eine Vielzahl an weiteren Möglichkeiten das Routing von Nachrichten zu beeinflussen. Im Rahmen dieser Arbeit sind die oben genannten Funktionen die einzigen, die relevant sind.

2.4 Usability Grundlagen

Im Deutschen wird Usability mit Gebrauchstauglichkeit übersetzt. Allerdings wird der Begriff im allgemeinen Sprachgebrauch nicht intensiv genutzt, daher wird im weiteren Verlauf der englische Begriff Usability benutzt.
Von der International Organization for Standards wird der Begriff Usability in der ISO EN 9241-11 [7] wie folgt definiert:

> "The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use."

Usability ist ein qualitatives Maß, das bewertet wie einfach ein Nutzer ein Nutzerinterface benutzen kann. Um ein hohes Maß an Usability sicherzustellen, muss Software so gestaltet werden, dass Nutzer ihre Ziele, die sie mit dem Programm erreichen wollen, effektiv, effizient und zu ihrer Zufriedenheit erreichen können. Diesem Prozess widmet sich das Usability Engineering. Um eine hohe Usability zu erreichen, gibt es diverse Verfahren und Methoden, die sich in Komplexität, Aufwand und nötiger Erfahrung der Anwender dieser Methoden teils stark unterscheiden.
Nach Sarodnick and Brau [11] werden diese Methoden in die Gruppen der empirischen, in denen Informationen durch die Befragung und Beobachtung der tatsächlichen Nutzer gesammelt werden, und der analytischen Methoden, in denen Usability Experten anhand von Richtlinien und Erfahrung mögliche Usability-Probleme aufdecken, eingeteilt.

Usability-Test Beim Usability-Test handelt es sich um ein empirisches Verfahren. In der Literatur sind auch die Begriffe Nutzertest oder Benutzbarkeitstest als Synonym zu finden. Bei den Usability-Tests wird wiederum zwischen summativen und formativen Tests unterschieden.
Formative Tests können früh im Entwicklungsprozess eingesetzt werden. Dafür kann mit Prototypen gearbeitet werden, die noch nicht die ganze Breite an Funktionen liefern oder mit auf Papier gemalten oder ausgeschnittenen Designs, sogenanntes Paper-Prototyping. [13] Diese Art Test deckt Usabilityprobleme im Design auf und hilft dabei Verbesserungen zu erarbeiten. Summative Tests dagegen können am Ende des Entwicklungsprozesses eingesetzt werden, um z. B. sicherzustellen, dass eine Anwendungen alle notwendigen Anforderungen erfüllt oder um zwei Anwendungen hinsichtlich ihrer Funktionalität und Usability zu vergleichen. [11]
Im Optimalfall beginnt der Prozess des Usability-Engineerings schon lange vor dem Design des Produktes [10, S.71f]. Häufig fehlt es Firmen allerdings an Zeit und Mitteln um umfassende Usability-Studien durchzuführen. Nielsen [10] schlägt vor die Methoden

Nutzertests, Vereinfachtes Lautes Denken und heuristische Evaluation priorisiert einzusetzen. Nutzertests müssen keine großen Studien mit vielen Nutzern sein. Nielsen[11] schreibt: Unter der Annahme, dass ein Nutzer 35% Prozent aller Usability-Probleme findet, erreichen 5 Nutzern eine Abdeckung von 85% aller Probleme, was für die meisten Anwendungen ausreichend ist. Mit wiederholten Tests kann diese Rate noch erhöht werden. Der Vorteil ist, dass ein Nutzertest mit nur 5 Teilnehmern, weniger Zeit und Geld kostet, als eine groß angelegte Studie mit 20-30 Teilnehmern.

Heuristische Evaluation Heuristische Evaluation ist eine weitere Methode die häufig eingesetzt wird um Usability-Probleme aufzudecken. Erfahrene Usabilityexperten bewerten anhand von Designrichtlinien die Usability der Anwendung. Diese Methode kann vor den Nutzertests durchgeführt werden um generelle Usability-Probleme aufzudecken. Aus Mangel an Usabilityexperten musste ich leider auf diese Methode verzichten. Nichtsdestotrotz habe ich versucht die Designrichtlinien der Heuristischen Evaluation zu berücksichtigen. [3, S. 62ff]
Eine komplette Auflistung dieser Designrichtlinien findet sich im Anhang auf Seite 36.

Durchführung von Nutzertests Für die Durchführung des Tests ist es sinnvoll einen Raum zu finden, in dem die Tester ungestört und ohne Unterbrechung den Test durchführen können. Dies kann ein speziell dafür eingerichtetes Usability Labor sein. Ein Büro in dem ungestört gearbeitet werden kann ist als Umgebung für kleinere Nutzertests jedoch ebenso geeignet. Bei häufiger Durchführung von Usability-Tests, sollte über die Anschaffung spezieller Hard- und Software nachgedacht werden. [3, S. 34f]

Lautes Denken Bei einem Test mit lautem Denken wird die Testperson gebeten während der Aufgabenbearbeitung laut auszusprechen was sie denkt. Dies beinhaltet Momente in denen die Testperson verwirrt war was ihr präsentiert wurde oder warum etwas passierte, Momente der Frustration, wenn etwas nicht funktionierte oder wenn etwas anderes eintrat als erwartet wurde, aber auch wenn etwas positiv auffällt oder etwas genau wie erwartet funktionierte.
Das laute Denken ist für viele Testpersonen unnatürlich und es fällt ihnen schwer ununterbrochen über das zu reden was sie gerade tun. „Was denken sie gerade“, „Was glauben sie ist dort zu sehen„ oder „Welche Reaktion haben sie erwartet“ sind nur ein paar Beispielfragen, die der Moderator des Tests stellen kann, um den Redefluss der Testperson wieder anzuregen [11, S. 164].
Durch das laute Denken kommt es auch dazu, dass die Testpersonen anders reagieren als würden sie stillschweigend arbeiten. Sie gehen meist langsamer und behutsamer vor, als normalerweise. Dadurch sind Metriken, die die Zeit pro Aufgabe beinhalten, nur bedingt nutzbar. Generell ist diese Art des Tests qualitativ und wird subjektiv durch die Person, die den Test auswertet, bewertet.

[11] Why you only need to test with 5 users, Jakob Nielsen, März 2000, `http://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/`

Obwohl die Aussagen der Methode des lauten Denkens mit die wertvollsten Informationen des Nutzertests darstellen, heißt das nicht, dass alle vorgeschlagenen Theorien und Lösungen der Testpersonen unreflektiert umgesetzt werden sollten. Es sollte verstärkt darauf geachtet werden, was der Nutzer macht und wo er hinguckt, wenn er auf ein Problem trifft. [10, S. 195ff]

Usability-Probleme Um die schwere der Usability-Probleme zu kategorisieren, kann eine simple 4-Punkte Skala, wie von [10, S. 103] vorgeschlagen, verwendet werden.

0: kein Problem Dies ist kein Usability-Problem.

1: Kosmetisch Der Nutzer ist kurzzeitig verwirrt, findet aber schnell wieder den Anschluss. Muss nicht behoben werde, es sei denn extra Zeit ist vorhanden.

2: Geringfügig Beheben des Fehlers, sollte geringe Priorität gegeben werden.

3: Schwerwiegend Der Nutzer ist frustriert, kann aber fortfahren. Wichtig, dass es behoben wird.

4: Katastrophe Nutzer kann nicht mit der Aufgabe fortfahren. Dieses Problem muss unbedingt behoben werden.

So wie ein Nutzer nicht alle Usability-Probleme aufdecken kann, kann auch ein einzelner Beobachter beziehungsweise Evaluator nicht alle Usability-Probleme identifizieren. Dieser sogenannte „Evaluator Effect“ wurde zuerst beschrieben durch [8]. Die Anzahl der Evaluatoren eines Nutzertests hat eine ähnliche Auswirkung wie die Anzahl an Testpersonen, im Bezug auf die Anzahl an entdeckten Usability-Problemen. Außerdem kann die Beurteilung der Schwere eines Problem zwischen zwei Evaluatoren ebenfalls unterschiedlich sein und kann zu einer schlechteren Priorisierung der Probleme führen. [10, S. 103] Es kann also empfehlenswert sein, nicht nur mehrere Tester testen zu lassen, sonder auch den einzelnen Test von mehreren Personen bewerten zu lassen.

Mentales Modell „A mental model is what the user believes about the system at hand.“[12]
Verschiedene Nutzer verfügen über unterschiedliche mentale Modelle. Zusätzlich kann auch das mentale Modell der Entwickler von dem der Nutzer abweichen. Dies kann zu einer Vielzahl von Usability-Problemen führen, wenn dem Entwickler dieser Umstand nicht bewusst ist.
Um Probleme, die das mentale Modell betreffen, zu beheben, gibt es zwei Möglichkeiten: (1) das System dem mentalen Modell der Nutzer anpassen. Das heißt, wenn die Mehrzahl der Nutzer eine Funktion an einer bestimmten Stelle erwartet, kann es von Vorteil sein diese Funktion dort anzubieten oder (2) das mentale Modell der Nutzer verbessern.

[12] Mental Models, Jakob Nielsen, Oktober 2010 – `http://www.nngroup.com/articles/mental-models/`

Hierbei können Funktionen besser erklärt oder Felder genauer beschriftet werden, damit die Nutzer schneller lernen, wie das Programm zu benutzen ist.
Hier zeigt sich ein weiterer Vorteil der Methode des lauten Denkens. Sie gibt einen Einblick in die mentalen Modelle der Nutzer und kann so helfen die Nutzer besser zu verstehen.

System Usability Score Der SUS-Fragebogen wurde entwickelt von Brooke [6] als eine Möglichkeit sich einen schnellen Überblick über die Usability eines Produktes zu machen. Die Fragen orientieren sich an der ISO 9241 und wurden nach der Likert-Skala formuliert. Das heißt, die Fragen sind als Aussagen formuliert, die entweder positiv oder negativ bewertet werden können. Die fünf Antwortmöglichkeiten reichen von „Stimme überhaupt nicht zu“ bis „Stimme voll zu“ und wir ein Wert von 0 bis 4 zugeordnet. Um den Score zu berechnen wird für ungerade Aussagen der Wert minus 1 genommen und für gerade Aussagen der der Wert von 5 abgezogen. Die Summe der so errechneten Werte wird mit 2,5 multipliziert und ergibt einen Wert für die System Usability zwischen 0 und 100. Der Test hilft dabei festzustellen ob Änderungen zwischen zwei Tests Auswirkungen auf die allgemeine Usability der Anwendung gehabt haben. Bangor et al. [2] haben über 2300 SUS-Fragebögen ausgewertet. Sie ermittelten einen Mittelwert aller SUS-Fragebögen bei 70.14 (s = 21.71). Auch wenn es schwer sei, konkrete Schlüsse aus den Werten zu ziehen, fassen sie zusammen, dass ein Wert von unter 50 beim SUS einen Grund liefert das Design nochmal zu überdenken, ein Wert von mindestens 70 auf ein immerhin ausreichendes Produkt hinweist und Werte von über 80 ein gutes Produkt darstellt.

3 Iterativer Entwicklungsprozess

Der erste Prototyp des Benachrichtigungssystems verfügte noch nicht über den vollen Funktionsumfang, um alle Anforderungen abzudecken. Zwischen den Tests wurde der Prototyp um zusätzliche Funktionen erweitert, die während der folgenden Tests evaluiert wurden. Nach jedem Nutzertest wurden die Ergebnisse analysiert, die schwerwiegendsten Probleme identifiziert und anschließend Anpassungen vorgenommen. Anhand der wiederholten Tests konnte festgestellt werden, ob die Änderungen die gewünschten Verbesserungen in der Usability mit sich brachten.
Die verschiedenen Nutzertests für das Benachrichtigungssystems wurden mit Wissenschaftlern des Botanischen Gartens und Botanischen Museums durchgeführt. Im Rahmen dieser Arbeit stand die Usability des Benachrichtigungssystems im Vordergrund. Es sollte ermittelt werden, ob Nutzer das Konzept der Abonnements verstehen. Dazu gehört das Erstellen und Löschen von Abonnements.
Da die Funktionen des Benachrichtigungssystems aber nur sinnvoll im Zusammenspiel mit dem Rest der Anwendung genutzt werden können, wurden während der Tests auch viele Probleme aufgedeckt, die zur Verbesserung der Usability der Anwendung im allgemeinen ausgewertet werden können. Durch diese Tests wurde zum ersten Mal Feedback von Nutzern außerhalb der AnnoSys-Gruppe gesammelt.

3.1 Entwicklungsstand vor der ersten Nutzerstudie

Um einen frühen Prototypen zu Verfügung zu stellen, mussten Einschränkungen im Funktionsumfang gemacht werden. Eine Einschränkung wurde beim Erstellen der Abonnements gemacht. Anstatt über Kriterien die Gruppe von Datensätzen einzuschränken, die abonniert werden soll, konnten nur einzelne Datensätze abonniert werden. Jedem Datensatz in AnnoSys wird eine Tripel-Id zugeordnet, die sich hierarchisch aus dem Kürzel des Instituts, der Sammlung zu der der Datensatz gehört und einer vom Institut vergebenen Objekt-Id zusammensetzt.
Beispiel-Id für einen Datensatz des Silbergrases am Botanischen Garten:

```
BGBM:Herbarium Berolinense:B 18 0014862
```

Diese Art von Abonnement war gut auf den Topic-Mechanismus von ActiveMQ abzubilden. Die Tripel-Id konnte als Name des Topics verwendet werden. Durch den hierarchischen Aufbau konnte beispielsweise die Objekt-Id weggelassen werden, um sich für eine ganze Sammlung anzumelden. Wenn der Nutzer mit der Id `123456` den oben genannten Datensatz abonniert, wurde die in Listing 2 dargestellte Camel Routendefition erzeugt. Eine Nachricht an dieses Topic wird dann an die Queue des Abonnenten weitergeleitet sowie an eine Camel Processor-Komponente übergeben, die die Nachricht mit Absender- und Empfängerinformationen versieht. Im Anschluss wird diese Nachricht an einen SMTP-Server geschickt.

Listing 2: Camel Routendefinition zum Abonnieren eines Datensatzes

```
from("activemq:topic:BGBM.Herbarium Berolinense.B 18 0014862")
  .to("activemq:queue:user.123456")
  .process(new UserMailProcessor("user.123456"))
  .to("mail://smtp.mymailserver.de?user=XXXXXX&password=
     XXXXXXX");
```

Über einen Doppelklick auf „Subscriptions“ in der Profilübersicht (siehe Abbildung 1 in Kapitel 2.2) kann die Abonnement-Übersicht geöffnet werden. Für den Doppelklick hatte ich mich entschieden, um konsistent mit dem Rest der Profilübersicht zu sein. Noch nicht veröffentlichte Annotationen konnten ebenfalls über einen Doppelklick geöffnet werden.

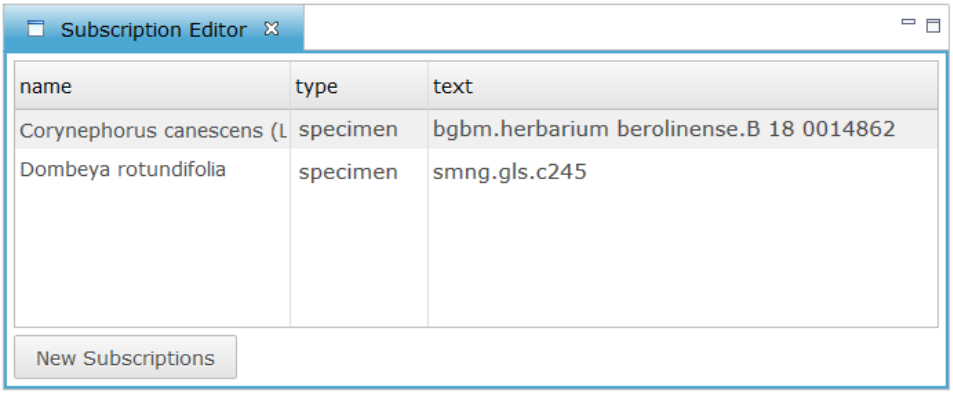


Abbildung 2: Abonement-Übersichtsseite

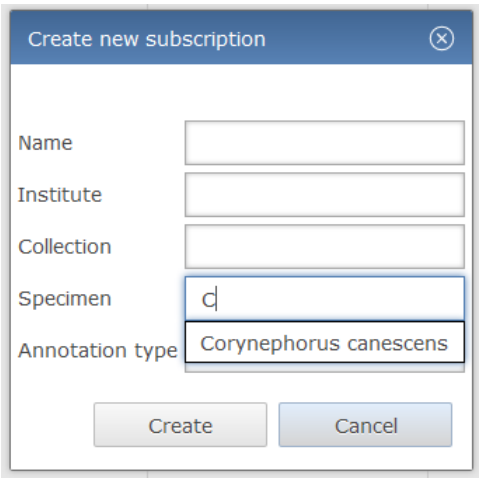


Abbildung 3: Dialog zum Erstellen eines Abonnements

Die Abonnement-Übersicht (Abbildung 2) zeigt in tabellarischer Form zu jedem Abonnement den wissenschaftlichen Namen und die Tripel-Id des abonnierten Datensatzes an. Die Spalte „type“ zeigt, ob das Abonnement für einen Datensatz, eine Sammlung

oder ein Institut erstellt wurde. Über die „New Subscription“-Schaltfläche können neue Abonnements mit Hilfe eines Dialogs (Abbildung 3) erstellt werden. In den verschiedenen Eingabefeldern des Dialogs werden mögliche Eingabewerte in einer Dropdown-Liste vorgeschlagen.
Eine weitere Einschränkungen des Prototypen war, dass Abonnements nicht gelöscht werden konnten. Sie wurden zwar aus der Oberfläche entfernt, aber Benachrichtigungen wären trotzdem verschickt worden. Außerdem konnten noch keine persönlichen Nachrichten an Nutzer geschrieben werden. Eine Schaltfläche, um die entsprechende Ansicht zu öffnen, existierte allerdings. So konnte dennoch getestet werden, ob die Testpersonen entdecken, dass sie anderen Nutzern Nachrichten schicken können.
Die Benachrichtigungsemail, die in Abbildung 4 zu sehen ist, zeigt den Stand der Email zum Zeitpunkt des dritten Tests. Außer, dass die Mail nach dem ersten Test übersetzt wurde, hat sich an ihr nichts geändert. Die Mail zeigt eine Annotation, die von Max Mustermann über die Aufsammlung des Belegs erstellt wurde. Er kommentiert, dass die GPS-Koordinaten vertauscht seien. Dies ist im Bereich „Annotationselement“ zu sehen. Der XML-Selektor gibt den Pfad des XML-Elements an, wo der Wert geändert werden soll. Diese Information ist speziell für Kuratoren wichtig, damit sie die Änderungen in ihre Datenbank übernehmen können.
Weitere Screenshots der kompletten Anwendung sind im Anhang auf Seite 34 zu finden.

Lieber AnnoSys-Nutzer,

vielen Dank für dein Interesse an Annotationen von Biodiversitätsdaten. Unten findest du die neu erstellte Annotation im Überblick.

Annotation

Name	Cakile maritima
GUID	BfN:FlorKart_BfN:558943
Annotierender	Max Mustermann
Annotationstyp	Gathering

Annotationselement

Elementname	Breitengrad (in Dezimalgrad)		
XML-Selektor	/DataSets/DataSet/Units/Unit/Gathering/SiteCoordinateSets/SiteCoordinates /CoordinatesLatLong/LatitudeDecimal		
	alter Wert	**neuer Wert**	**Erwartung**
	48.16	11.58	Update
Elementname	Längengrad (in Dezimalgrad)		
XML-Selektor	/DataSets/DataSet/Units/Unit/Gathering/SiteCoordinateSets/SiteCoordinates /CoordinatesLatLong/LongitudeDecimal		
	alter Wert	**neuer Wert**	**Erwartung**
	11.58	48.16	Update

Du kannst dir die Annotation auch online im AnnoSys-System angucken: http://example.com /AnnoSys?repositoryURI=http://example.com/services/annotations/BGBM/AnnoSys/1396331081954

Du erhältst diese Mail, weil du entweder eine Annotation erstellt hast oder eines deiner Abonnements auf den Datensatz dieser Annotation zutreffen.

Viele Grüße
dein AnnoSys-Team

Abbildung 4: Emailbenachrichtigung einer Annotation

3.2 Erster Test

Für den ersten Test wurden fünf Mitarbeiter des BGBMs eingeladen. Eine Person musste den Test leider kurzfristig absagen, sodass der erste Test nur mit vier Personen durchgeführt werden konnte. Unter den Testpersonen befanden sich sowohl Frauen als auch Männer. Wenn im Folgenden von dem Tester oder der Testperson die Rede ist, ist sowohl die Testerin als auch der Tester gemeint. Eine Bedingung an die Testpersonen war, dass sie Wissen über Herbarbelege und das Vorgehen des physischen Annotierens mitbrachten, um zu verstehen, wofür AnnoSys benutzt wird. Testperson 1 und 2 waren Informatiker und Testperson 3 Ökologe. Sie beschäftigen sich eher mit Datenverarbeitung und der Entwicklung von Standards anstatt der direkten taxonomischen Arbeit mit Herbarbelegen. Die vierte Testperson ist Taxonom und die einzige Testperson, die zur Zielgruppe der Anwendung gehörte. Außerdem ist sie mit der Datenbankenverwaltung von Herbardaten vertraut, somit ist ihr das abcd-Datenformat nicht unbekannt.
Alle Testpersonen haben angegeben, dass ihnen die Anwendung im Rahmen eines Vortrages bereits vorgeführt wurde. Praktisch eingesetzt hatten sie die Anwendung zuvor jedoch nicht. Allerdings war Tester 4 in den Prozess der Anforderungsanalyse im Jahr 2012 eingebunden und hatte daher Vorstellung wie das Programm zu benutzen sei.
Der Test wurde auf meinem Laptop und einem externen Monitor (22") durchgeführt. Auf dem externen Monitor wurde die AnnoSys-Anwendung in Firefox angezeigt und auf dem Laptopmonitor eine Trashmail-Webseite[13], an die die Benachrichtigungsmails verschickt wurden.
Dieser Aufbau wurde gewählt, um das Benachrichtigungsverhalten einer Email-Anwendung zu simulieren. Bei Email-Anwendungen wie Microsoft Outlook ist es üblich, dass Benachrichtigungen auf dem Bildschirm erscheinen, wenn eine neue Mail eingeht. Bei diesem Trash-Mail Anbieter wird die Webseite sofort aktualisiert, wenn eine neue Mail vorhanden ist.
Während des Tests wurde ein sogenannter Screencast aufgenommen. Das heißt, es wird der externe Monitor und via Mikrofon die Testperson für spätere Analysen aufgenommen.
Der Ablauf des Tests war in drei Teile gegliedert. Als Erstes erläuterte ich, warum dieser Test durchgeführt wird und was sie auf den einzelnen Monitoren sehen. Dabei wurde die Methode des lauten Denkens erklärt. Außerdem wurden die Tester darüber aufgeklärt, dass sie aufgenommen werden.
Der zweite Teil war der eigentliche Test. Er beinhaltete folgende Aufgaben[14]:

1. Annotation erstellen
2. Konkreten Datensatz abonnieren
3. Abonnement löschen
4. Annotation suchen
5. Nachricht schreiben
6. Mail lesen

[13] www.byom.de
[14] Der ausführliche Aufgabentext ist im Anhang auf Seite 38

7. Nachrichten lesen

Die Aufgaben wurde in dieser Reihenfolge bearbeitet. Zu jedem Zeitpunkt lag ihnen nur der für die jeweilige Aufgabe relevanter Aufgabentext vor. Am Ende folgte ein informelles Interview, in dem wir Probleme und Anmerkungen besprachen.

3.2.1 Usability-Probleme

Im ersten Test identifizierte ich 18 Probleme. Davon beziehen sich 12 auf die Anzeige und Erzeugung von Abonnements. Die restlichen 6 Probleme betreffen andere Teile der Anwendung. In Tabelle 2 werden die Probleme aufgelistet und auf ihre Schwere beurteilt. Die Probleme wurden nach den Bereichen der Anwendung gruppiert, in denen sie auftraten. Die Benennung der Schlüssel spiegelt diese Gruppierung wieder: „Abo“ für Probleme, die das Erzeugen, Löschen und Anzeigen der Abonnements betreffen, „Nach“ für Probleme, die Nachrichten betreffen, „Mail“ für Probleme mit der E-Mail und „Anno“ für Probleme, die das Erstellen von Annotationen betreffen.
Wie in Kapitel 2.4 beschrieben, wird die Fatalität der Fehler auf einer Skala von 0 bis 4 bewertet, wobei 0 kein Problem entspricht und 4 einem katastrophalem Problem.

Schl. Schlüssel des Problems

Anzahl der Auftritte des Problems

Fat. Fatalität des Problems

Tabelle 2: Gefunden Usability-Probleme des ersten Tests

Schl.	Beschreibung	#	Fat.	Kommentar
Abo1	Konnte Abonnement-Übersicht nicht öffnen	2	4	Der erwartete Ablauf war anders. Erst Datensatz öffnen, dann „hier anmelden“ klicken
Abo2	Doppelklick auf Subscriptions nicht offensichtlich	4	4	
Abo3	Abonnement-Übersicht nicht lokalisiert	2	2	
Abo4	Dialog zum Erstellen war unklar	2	4	Unklar, was in die einzelnen Felder eingegeben werden soll
Abo5	Keine Möglichkeit Abonnements in der Abonnement-Übersicht zu löschen	2	2	
Abo6	NewSubscriptionButton am unteren Rand nicht gesehen	1	1	

Tabelle 2 – fortgesetzt von voriger Seite

Schl.	Beschreibung	#	Fat.	Kommentar
Abo7	Unklar was eine Subscription sein soll	2	3	
Abo8	Überrascht, dass schon Abonnements da waren	1	2	Nicht ausreichend darüber informiert, was nach dem Annotieren passiert
Mail1	Anrede als Subscriber unklar.	1	1	Nutzer war verwirrt, warum er überhaupt ein Subscriber ist
Mail2	Mail war nicht deutsch lokalisiert	1	2	Keiner der Tester hat einen Kommentar gemacht, dass die Mail auf englisch ist.
Nach1	Sortierung der Nachrichten ist aufsteigend	1	2	
Nach2	Keine Indikation auf neue Nachrichten	1	2	
Anno1	Allgemeines Kommentarfeld wird als erster Impuls benutzt	2	2	Verleitet dazu die Eingabemaske zu ignorieren.
Anno2	Inhalt des alten Wertes in der Tabellenspalte komplett anzeigen	1	2	Der Nutzer suchte nach dem kompletten Namen in den Anzeigefeldern, hat das entsprechende Feld aber nicht gefunden, weil der Name abgeschnitten war
Anno3	Was ist eine Erwartung	2	1	
Anno4	Alten Wert automatisch beim selektieren in die Neuer-Wert-Spalte eintragen	1	1	
Anno5	Keine Information über Erfolg oder Misserfolg	3	3	
Anno6	Speichern/Veröffentlichen war schwierig	4	4	Obwohl alle darauf gekommen sind Publish zu klicken, war niemand sicher, dass es das Richtige ist.

Tabelle 3 zeigt welche Aufgaben wie häufig erfolgreich bearbeitet werden konnten. Es ist deutlich zu sehen, dass die Abonnements im Gegensatz zum Rest der Anwendung große Probleme verursachten.
In Aufgabe 6 sollten die Testpersonen nur ihre Meinung zu der angezeigten Mail abgeben.

Daher wurde diese Aufgabe nicht mit Erfolg oder Misserfolg bewertet.

Tabelle 3: Erfolgreich durchgeführte Aufgaben des ersten Tests

Aufgabe	Erfolg	Misserfolg
1 Annotation erstellen	4	0
2 Abonnement erstellen	1	3
3 Abonnement löschen	2	2
4 Annotation finden	4	0
5 Nachricht schreiben	4	0
6 E-Mail lesen	-	-
7 Nachrichten lesen	4	0

Die gesamte Bewertung von Nutzertests ist sehr subjektiv und stark vom Beobachter abhängig. Es kann von Beobachter zu Beobachter unterschiedlich sein, was als Problem angesehen wird und mit welcher Schwere ein gefundenes Problem bewertet wird. Daher ist es auch schwierig quantitative Bewertungen abzugeben.
In diesem Test hat sich der Doppelklick zum Öffnen der Abonnementübersicht als katastrophales Problem herausgestellt. Generell war die Abonnementübersicht problematisch. Zwei von vier Testern konnten sie durch den Doppelklick auf Subscriptions nicht öffnen. Sie konnte nur von Testern geöffnet werden, die schon vorher, ohne ersichtlichen Grund, alles „gedoppelklickt“ haben. Die anderen Tester mussten darauf aufmerksam gemacht werden, um fortfahren zu können.
Im nächsten Schritt konnten zwei Nutzer keine neuen Abonnements erstellen. Tester 1 hat den „New Subscription“-Button am unteren Rand des Fensters nicht entdeckt.
Testperson 2 war unklar, welche Angaben im Dialog gemacht werden sollen.
Alle Nutzer machten während des Tests die Aussage, dass sie lieber erst einen Datensatz öffnen wollen, um von dort direkt den Datensatz zu abonnieren.
Das Doppelklickproblem kann auf zwei Gründen beruhen. Erstens waren die Teilnehmer vermutlich nicht gewohnt innerhalb des Browsers einen Doppelklick auszuführen und zweitens haben sie nicht erwartet, dass innerhalb eines Baumes ein Doppelklick etwas anderes bewirkt als den Baum aufzuklappen.

> „Das ist sehr unüblich. Ein Doppelklick auf einen Baum öffnet sonst nur den Baum.“
>
> – Tester 2

Die Probleme Abo3, Abo4, Abo7, Mail1 und Mail2 beziehen sich auf Probleme, die mit dem Verständnis der Beschriftung zu tun haben. Eine Übersetzung der Beschriftungen könnte schon zu einer Verbesserung führen.
Obwohl folgendes nicht dem Benachrichtigungssystem zugeordnet wird, sollte es dennoch erwähnt werden, da es bei allen Testpersonen zu Problemen führte. Alle Tester haben relativ lange gesucht, um ihre Annotation zu speichern bzw. zu veröffentlichen (Anno6).

Sie haben nur auf den Button in der Toolbar, der mit dem Tooltip[15] „Veröffentlichen“ versehen war, geklickt, weil es das einzige war, was die Tester irgendwie in Verbindung mit „Speichern“ bringen konnten.

> „Ich klick' hier jetzt rauf, aber ich habe keine Ahnung, was das macht.“
>
> – Tester 1

3.2.2 Fazit

Wie oben beschrieben, war es nicht möglich die Benachrichtigungen komplett losgelöst vom Rest der Anwendung zu untersuchen. Viele der gefunden Probleme betreffen auch das Annotieren von Datensätzen und das allgemeine Layout bzw. Bedienung des Systems. Da die Anwendung noch nie von echten Nutzern getestet wurde, war das aber auch zu erwarten. Zusätzlich wurden diese Probleme auch aufgenommen und ein paar von ihnen bis zum nächsten Test behoben, um einen flüssigeren Arbeitsablauf zu gewährleisten.
Obwohl viele Probleme gefunden wurden, war die Wahl der Teilnehmer nicht optimal. Wir sind davon ausgegangen, dass alle Mitarbeiter des BGBM, auch wenn sie nur entfernt mit Herbarbelegen arbeiten, als Testpersonen geeignet sind. Im Bezug auf die Zugänglichkeit der Benachrichtigungen war das kein Problem, wurde aber beim Inhalt deutlich. Testperson 4, der Tester mit taxonomischen Hintergrund, zeigte ein vorsichtigeres und detaillierteres Vorgehen im Test als alle anderen. Er hatte genauere Vorstellungen was ihn erwartet und wie Probleme gelöst werden sollten. Für den nächsten Test haben wir dann darauf geachtet mehr Teilnehmer zu akquirieren, die entweder Kuratoren von Herbarien, Taxonomen oder Botaniker sind.
Testperson 4 hatte sich auch postiv über die Informationen in der Email geäußert. Seiner Meinung ist die Anzeige der XML-Selektoren als sinnvoll anzusehen. Die anderen Testpersonen äußerten sich nicht zum Inhalt der Emails.

3.3 Erste Iteration

Im Vordergrund dieser Iteration stand, den Funktionsumfang der Abonnements zu erweitern. Zusätzlich wurden für die Probleme mit höchster Fatalität und Häufigkeit, die den Ablauf unmöglich machten oder erheblich beeinträchtigten, Lösungen erarbeitet und implementiert.

3.3.1 Individuelle Abonnements

Aufgrund der vielen Probleme beim Erstellen von Abonnements im ersten Test wurde das Userinterface komplett überarbeitet.
In der neuen Version kann über ein Freitexteingabefeld und DropDown-Menü eine Vielzahl an Kriterien hinzugefügt werden, auf die die Abonnements zutreffen sollen (siehe

[15] Ein Tooltip ist, in diesem Zusammenhang, der Text, der angezeigt wird, wenn die Maus über ein Element gehalten wird.

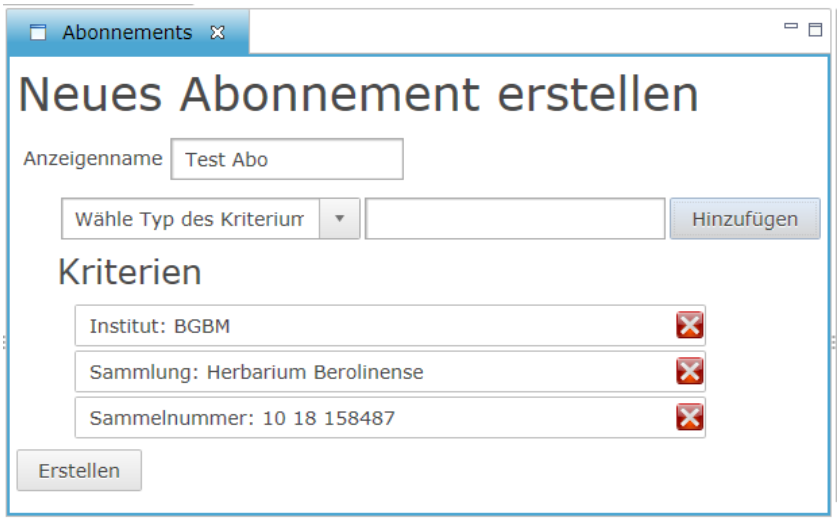


Abbildung 5: Neue Ansicht zum Erstellen von Abonnements

Abbildung 5). Die einzelnen Kriterien werden mit einem logischen „und“ verknüpft. Kriterien können auch wieder entfernt werden, solange noch nicht auf Erstellen geklickt wurde.
Die Abonnements im ersten Teil, bezogen sich nur auf Daten, die schon in der Tripel-Id der Datensätze enthalten sind. Weil die Tripel-Id als Adresse der jeweiligen Topic-Endpoints genutz werden konnte, ließ sich so komfortabel das entsprechende Routing einrichten. Das war für die individuellen Abonnements nicht mehr möglich. Es muss zusätzlich der Datensatz überprüft werden, ob die Kriterien des Abonnements auf den Datensatz zutreffen.
Dafür habe ich zwei Lösungsansätze erarbeitet.

Ansatz 1 Abonnements werden als POJOs (Plain Old Java Objects) modelliert und in einer SQL-Datenbank persistent gespeichert (siehe Abbildung 6). Sie enthalten eine Liste von Kriterien, die auf einen bestimmten Datensatz zutreffen müssen. Wird nun eine Annotation erstellt, werden alle Abonnements durchsucht und überprüft, ob die Kriterien des Abonnements auf die Annotation und den dazugehörigen Datensatz passen. Ist das der Fall, wird über Camel dem Nutzer, dem das Abonnement gehört, eine Nachricht an seine Queue gesendet. Camel kümmert sich, wie schon vorher, um das Versenden der entsprechenden Mail. Dieser Ansatz ist relativ simpel und scheint ohne viel Aufwand implementierbar zu sein. Im Prinzip wird also ein eigenes Anmelde-Veröffentlichen-Verfahren implementiert und Camel zu einer Art Mailbox System für die Nutzer degradiert.

Ansatz 2 Ein zweiter Ansatz benutzt weiterhin Camel, um Routen zwischen Datensatz und Nutzern zu definieren.
Wird ein neues Abonnement erstellt, werden die Kriterien des Abonnements für alle bekannten Datensätze überprüft. Für jeden Datensatz auf den die Kriterien zutreffen

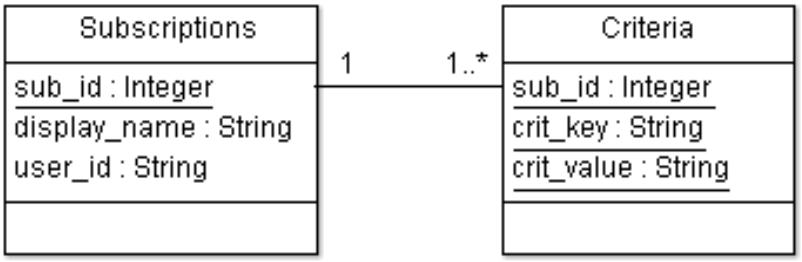


Abbildung 6: Datenbankschema eines Abonnements

wird für das entsprechenden Camel-Topic eine neue Route definiert. Der Vorteil ist, dass nur eine Nachricht an das entsprechende Topic gesendet werden muss und Camel sich um die Verteilung an die jeweiligen Nutzer kümmert.
Immer wenn ein neuer Datensatz ins AnnoSys-System importiert wird, wird für jedes Abonnement überprüft, ob es auf den neuen Datensatz passt und eine entsprechende Route hinzugefügt. Es gibt aber in Camel keinen einfachen Weg, um herauszufinden für welchen Nutzer welche Routen definiert wurde. Ganz davon abgesehen, dass die Routen nicht widerspiegeln, auf welche Abonnementkriterien sie beruhen. Daher müssten wie im ersten Lösungsansatz die Abonnements persistent gespeichert werden um später die entsprechenden Routen wieder zu löschen und um die Kriterien in der Anwendung anzuzeigen.

3.3.2 Änderungen

Am Ende habe ich mich für die erste Methode entschieden. Schon für die erste Version, nahm es mehr Zeit in Anspruch als erwartet Camel zuverlässig in die Anwendung zu integrieren. Zusätzlich ist für beide Ansätze ein persistenter Speicher notwendig um die Kriterien zu speichern. Daher empfand ich es als sinnvoller und den Aufwand als überschaubarer, ein eigenes Verteilsystem zu implementieren, anstatt das Routing Camel zu überlassen. Da das Mail verschicken und Nachrichten aus Camel auslesen bereits funktionierte, beließ ich es dabei.
In der Profilübersicht können im Bereich Abonnements diese samt Kriterien angezeigt und das komplette Abonnement auch wieder gelöscht werden.
In Tabelle 4 werden alle weiteren Änderungen aufgelistet. Die erste Spalte der Tabelle bezieht sich auf das Usabilityproblem, das mit dieser Änderung behoben werden sollte. Die als sonstige gelisteten Änderungen wurden vorgenommen, ohne dass ein Usability-problem dazu dokumentiert wurde.

3.4 Zweiter Test

Wie zuvor in Kapitel 3.2.2 erwähnt, hatte sich herausgestellt, dass die ausgewählten Tester nicht zur eigentlichen Zielgruppe der Anwendung gehörten. Beim zweiten Test wurden die Tester dann mit mehr Sorgfalt ausgewählt, um bessere Repräsentanten der

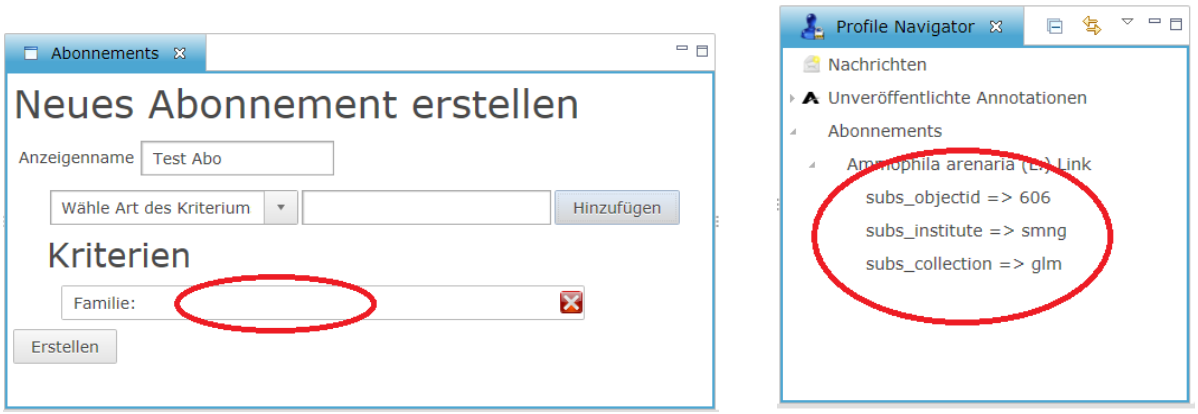
Tabelle 4: Änderungen der ersten Iteration

Schl.	Beschreibung
Abo1	Button in Toolbar zum Erstellen von neuen Abonnements hinzugefügt
Abo2	Doppelklick öffnet den Baum
Abo4/5/6	Neues System, um Abonnements zu erstellen
Abo7	„Subscriptions“ überall in „Abonnement“ geändert
Abo8	Ein Infodialog informiert darüber, dass nach dem Annotieren ein Abonnement erstellt wurde
Mail1	Text der Mail geändert, sodass darüber informiert wird, warum der Nutzer eine Mail bekommt.
Mail2	Mail ins Deutsche übersetzt
Nach1	Nachrichten werden abfallend sortiert
Anno5	Infodialog nach dem Annotieren wird angezeigt (siehe Subs8)
Anno6	Toolbaricons mit Text versehen
sonstige Änderungen	
	Alle Bereiche der Abonnements ins Deutsche übersetzt.
	Ansicht zum Schreiben einer persönlichen Nachricht hinzugefügt

AnnoSys-Nutzer zu finden. Der Test wurde mit fünf Testern durchgeführt, die alle eine taxonomische Ausbildung haben und zur Zeit auch als solche arbeiten oder bis vor kurzem lange in dem Bereich gearbeitet haben. Alle Testpersonen haben nach dem Test angegeben, dass sie glauben, sie gehören zur Zielgruppe der Anwendung. In diesem Aspekt wurde der Test durch die bessere Auswahl repräsentativer. Dies hat eventuell dazu beigetragen, dass mehr Usability-Probleme gefunden wurden.

3.4.1 Usability-Probleme

Im zweiten Test habe ich 37 Probleme identifiziert, von denen 17 den Benachrichtigungen und Abonnements zugeordnet werden konnten und die restlichen 20 dem Annotierungsprozess. Tabelle 5 listet alle gefundenen Probleme auf.
Für die Benachrichtigungen wurden drei Probleme als katastrophal bewertet und müssen auf jeden Fall behoben werden. Zum einen können Abonnements erstellt werden, die leere Kriterien beinhalten (Abo4), dieser Zustand ist in Abbildung 7a zu sehen, und zum Anderen werden in der Abonnementübersicht nicht die korrekten Beschriftungen der Kriterien angezeigt (Abo10, siehe Abbildung 7b). Erstaunlicherweise führte das dazu, dass auch keine Nachrichten und Emails verschickt wurden (Abo7).
Die Probleme, die mit schwerwiegend bewertet wurden, beziehen sich zum großen Teil darauf, dass die Tester nicht verstanden haben, was Kriterium im Bezug auf ein Abon-



(a) (b)

Abbildung 7: (a) Abonnement mit leerem Kriterium, (b) Falsche Werte der Kriterien in der Profilübersicht

nement bedeuten soll. Allerdings haben alle Tester im Interview angegeben, dass sie es nicht als schlimm empfanden und sie vermutlich beim nächsten mal wüssten, wie es richtig zu benutzen sei.

> „Eine gewisse Einarbeitungszeit ist bei neuen System, ja immer von Nöten."
> – Tester 2

Obwohl die Icons in der Toolbar[16] eine Beschriftung erhalten haben, hat keiner der Tester sofort den Speichernbutton gefunden. Nachdem sie den Button fanden, gab es keine Unklarheiten bezüglich seiner Funktion. Dies ist eine Verbesserung, im Gegensatz zum ersten Test als der Button noch mit „Veröffentlichen" beschriftet war.
Selbst wenn die Tester bei einer Aufgabe nicht weiterkamen, hat niemand versucht die Hilfe in der Toolbar zu benutzen. Im Interview hat ein Tester explizit gesagt, er würde so etwas nicht benutzen.

Tabelle 5: Gefunden Usability-Probleme des zweiten Tests

Schl.	Beschreibung	#	Fat.	Kommentar
Abo1	Werte in der Dropdown-Box sind schlecht sortiert.	1	1	
Abo2	Unklar was ein Kriterium ist.	3	3	1. Kann es sich nach gewisser Zeit über die Auswahl in der Dropdown-Box herleiten. 2. Zögert was es sein soll

[16]Die Toolbar ist die oberste Leiste der Anwendung die mehrere Buttons zeigt um z. B. die Hilfe zu öffnen, Annotationen zu speichern oder neue Abonnements zu erstellen

Tabelle 5 – fortgesetzt von voriger Seite

Schl.	Beschreibung	#	Fat.	Kommentar
Abo3	Label „Anzeigename“ unklar.	2	3	Obwohl unklar, was es bedeutet, gibt ein Tester trotzdem einen guten Wert ein
Abo4	Es kann ein Kriterium ohne Wert hinzugefügt werden	2	4	
Abo5	Objekt-Id missverständlich, weil nicht gebräuchlich	1	2	
Abo6	Tester wünscht sich größere Auswahl an Kriterien	1	2	
Abo7	Möchte Abonnements im Zusammenhang mit dem Datensatz sehen	2	2	Vielleicht Dialog anbieten, ob zu einem geöffneten Datensatz ein Abonnement eingerichtet werden soll
Abo8	Nachricht werden nicht verschickt	1	4	
Abo9	Möchte beim Annotieren gefragt werden, ob ein Abonnement eingerichtet werden soll.	1	2	
Abo10	Übersicht in den Kriterien zeigen Datenbankwerte an	2	4	
Abo11	Hinzufügen und Erstellen wirkt doppelt	1	2	
Nach1	Text „Nachricht senden“ wird nicht komplett angezeigt	3	2	Tester haben das nicht kommentiert, ist mir aber aufgefallen
Nach2	Nicht sofort „Nachricht senden“ gesehen. Schlägt Icon vor.	2	3	
Nach3	Erwartet gesendete Nachrichten auch zu sehen.	1	2	
Nach4	Sprache der Mail einstellbar machen	1	2	
Nach4	Gibt keinen Use-Case für Nachrichten schreiben	1	4	
Mail1	XML-Selektoren in Mail nicht notwendig	1	3	
Einst1	Klick auf Einstellungen zeigt Fehlermeldung	1	4	

Tabelle 5 – fortgesetzt von voriger Seite

Schl.	Beschreibung	#	Fat.	Kommentar
Anno1	Tester möchten den wiss. Namen via Selektion mit Maus annotieren	1	0	
Anno2	Annotationstyp für Namensänderung/Synonym fehlt	1	0	
Anno3	Erwartet, dass sein Name sofort als Annotierender eingetragen wird	1	1	
Anno4	Bei Eingabe vom kompletten wiss. Namen soll der geparsed und in die entsprechenden Felder eingefügt werden.	2	1	
Anno5	Speichern wird lange gesucht	3	2	
Anno6	Tester erwartet Bestätigung/Übersicht über seine Annotation bevor sie veröffentlicht wird.	1	2	
Anno7	Spalten missverständlich beschriftet	1	2	
Anno8	Doppelklick führt zu Verzögerungen	1	4	
Anno9	Feld „Erwartung“ unklar	3	4	
Anno10	Remove missverstanden	1	4	
Anno11	Remove und Neuer Wert kann gemeinsam ausgewählt werden	1	4	
Anno12	Tabelle verändert Größe nicht	3	4	
Anno13	Annotationstyp nicht überall lokalisiert (ProfilNavigator und Mail)	1	2	
Anno14	Möchte in der Annotationsansicht ein Kommentar zur Annotation eingeben	1	2	
Suche1	Suchresultate verringern den Platz für Annotationseditor zu stark	1	4	

Tabelle 5 – fortgesetzt von voriger Seite

Schl.	Beschreibung	#	Fat.	Kommentar
Suche2	Einfacheren Weg, um alle Annotationstypen auszuwählen	1	3	
Suche3	Resultatsbaum nicht aufgeklappt	1	4	
Suche4	Freitextsuche gewünscht	1	0	

3.4.2 Fazit

In Tabelle 6 ist zu sehen welche Aufgaben erfolgreich durchgeführt wurden. Die Aufgaben sind in der Reihenfolge dargestellt in der sie im Test am ersten Tag durchgeführt wurden. Im Bezug auf die Abonnements habe ich hier einige Aufgaben als Teilerfolg bewertet. Das bedeutet, dass die Testpersonen zwar eine Annotation erstellt haben, aber nicht die Korrekte. So wurde z. B. in Aufgabe 4 nicht die Triple-Id verwendet, sondern der Artname, um den Datensatz zu bestimmen. Zwei Testpersonen konnten Aufgabe 2 „Abonnement für eine konkreten Datensatz erstellen“ nicht erfolgreich bearbeiten. Diesen Testpersonen wurde dann Aufgabe 4 „Abonnement für eine Familie erstellen“ gar nicht gestellt, weil sie einen ähnlichen Arbeitsablauf erforderten. Daher sind in der Zeile zu Aufgabe 4 nur 3 Testpersonen angegeben. Die Lösung der Aufgabe wäre gewesen, über das Kontextmenü in der Annotationsübersicht einen Datensatz zu abonnieren (siehe Abbildung 8) oder alternativ in der Abonnementansicht die Kriterien der Tripel-Id manuell zusammenzustellen.
Dieser Test war aus terminlichen Gründen auf zwei Tage aufgeteilt. Dies gab mir die Möglichkeit den Ablauf des Tests für den zweiten Tag zu überdenken. Für die restlichen drei Testpersonen wurde die Reihenfolge der Aufgaben geändert. Aufgabe 3 und 4 wurden vor Aufgabe 2 durchgeführt. Dies sollte den Testern die Möglichkeit geben, während Aufgabe 3 „Annotation suchen“, sich mit der Annotationsübersicht vertraut zu machen und dort zu entdecken, dass sie Datensätze abonnieren können.

Tabelle 6: Erfolgreich durchgeführte Aufgaben des zweiten Tests

Aufgabe	Erfolg	Teilerfolg	Misserfolg
1 Annotation erstellen	4	0	1
2 Abonnement Familie	3	0	2
3 Annotation suchen	3	0	2
4 Abonnement konkreter Datensatz	0	3	0
5 Abonnement löschen	5	0	0
6 Nachricht schicken	5	0	0
7 Mail anschauen	-	-	-

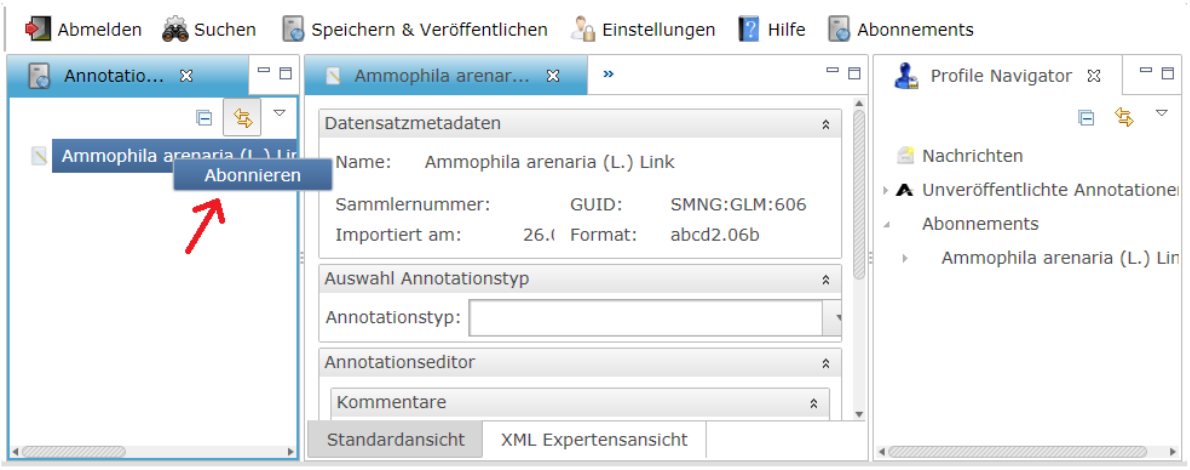


Abbildung 8: Ein Abonnement für einen Datensatz kann über ein Kontextmenü auf den Datensatz erstellt werden

3.5 Zweite Iteration

Um noch einen dritten Test durchführen zu können, musste die zweite Iterationsphase kürzer ausfallen. Daher beschränken sich die Änderungen auf die katastrophalen Probleme und kleinere kosmetische Änderungen, die helfen sollten, Abonnements und Kriterien besser zu verstehen.

Tabelle 7: Änderungen nach dem zweiten Test

Schl.	Beschreibung
Abo2	Hilfe-Icon in Abonnementansicht hinzugefügt
Abo3	Anzeigename in Abonnementname geändert
Abo4/8	Leeres Kriterium nicht mehr möglich
Abo11	Hinzufügen durch Plus-Icon geändert
Abo10	Kriterien zeigen richtigen Text in der Übersicht
Anno11	Keine Werte mehr einfügbar in Löschen-Annotationen
sonstige Änderungen	
	Ungelesene Nachrichten werden durch fetten Text hervorgehoben
	Beschriftungen von Abonnementübersicht und Profilübersicht geändert

Im zweiten Test war drei Testpersonen unklar, was ein Kriterium sein soll (siehe *Abo2*). In Übereinstimmung mit den Heuristiken, dass es sinnvoll sein kann eine Hilfe anzubieten, wurde ein Icon hinzugefügt, durch das ein erklärender Text zu den Abonnements und Kriterien angezeigt werden kann.

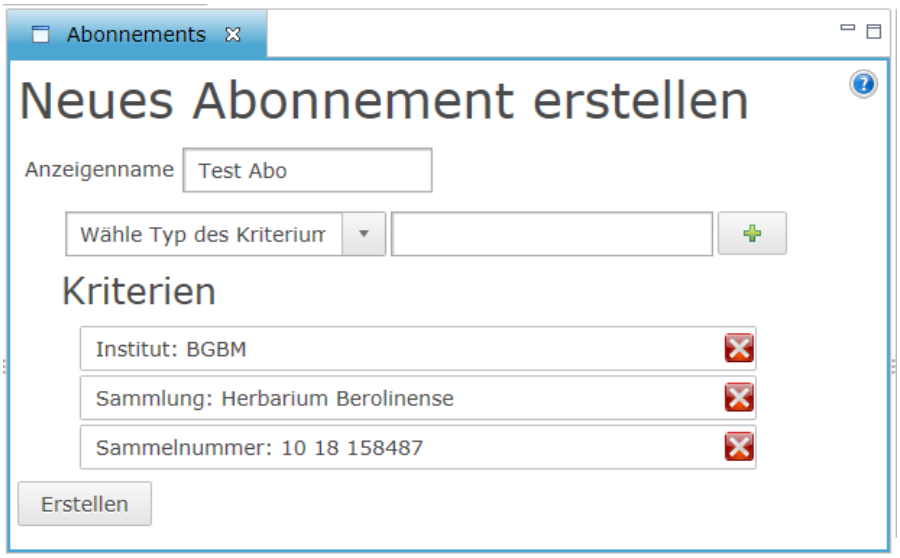


Abbildung 9: Abonnement-Editor nach der zweiten Iteration. Es wurde ein Hilfe-Icon hinzugefügt und der Text des Hinzufügen-Button durch ein Plus-Icon ersetzt.

3.6 Der dritte Test

Der dritte Test wurde nur mit drei Testpersonen durchgeführt, da es immer schwieriger wurde geeignete Wissenschaftlern des BGBMs, die ausreichend mit der taxonomischen Arbeit vertraut sind, zu finden und mit diesen zeitnah Termine zu vereinbaren.
In diesem Test wurde auf die Aufgabe „Nachricht verschicken“ verzichtet, da sie bereits im zweiten Test von allen Testpersonen erfolgreich bearbeitet wurde und keine Usability Probleme aufgetreten sind. Die restlichen Aufgaben wurden in der selben Reihenfolge gestellt, wie den Testpersonen des zweiten Tests am zweiten Tag.

3.6.1 Usability-Probleme

Insgesamt habe ich im dritten Test 20 Probleme identifiziert von denen 8 Abonnements und die Benachrichtigungsmails betrafen. Das ist erheblich weniger als im zweiten Test, was damit zusammenhängen kann, dass mit weniger Testpersonen getestet wurde.
In diesem Test habe ich keine Probleme als katastrophal bewertet. Als schwerwiegend wurde zum Beispiel Abo4 bewertet. Zwei Tester haben lange gebraucht um auf den Plus-Button zu klicken. Statt dessen wollten sie immer das Kriterium mit Eingabe bestätigen. Usability Problem Abo2 bezieht sich auf die Benennung der Kriterien. Um dieses Problem zu lösen muss noch einmal genau ermittelt werden, welche Sprache die Nutzer hier erwarten. In diesem Zusammenhang könnte auch recherchiert werden welche Kriterien zur Auswahl am besten hinzugefügt werden sollten.

Die Anzahl der Misserfolge, die in Tabelle 9 zu sehen sind, lässt immer noch bestehendes Verbesserungspotential schließen.

Tabelle 8: Usability-Probleme des dritten Tests

Schlüssel	Beschreibung	#	Fat.
Abo1	Tester war nicht klar, dass ein Abonnement beim Annotieren eingerichtet wird	1	2
Abo2	Tester empfindet einige Namen der Kriterien als unklar für Biologen	1	3
Abo3	Unklar was ein Kriterium sein soll	1	2
Abo4	Mit „Eingabe“-Taste Kriterium hinzufügen	2	3
Abo5	Überrascht als ein neues Kriterium erscheint, wenn er den Plus-Button anklickt.	1	3
Mail1	XML Selektor in der Mail nicht hilfreich	1	2
Mail2	Mail sei allgemein unübersichtlich	1	2
Mail3	Manche Begriffe erscheinen zu technisch und für Biologen nicht nachvollziehbar	1	3
Anno1	Anzeige des Formats ist unklar auf was sich das bezieht	1	2
Anno2	Tester möchten den wiss. Namen via Selektion mit Maus annotieren	1	0
Anno3	Lange nach Speichern-Button gesucht	3	4
Anno4	Feld „Erwartung“ unklar	3	4
Anno5	Toolbar nicht als Teil des Programms gesehen	1	0
Anno6	„Wert“ sei nicht Biologenjargon	1	2
Xml1	Expertenansicht zu unkomfortabel	1	2
Suche1	Zu viele Eingabefelder in der Suchmaske	1	3
Suche2	Erwartet eine Art Autovervollständigung für Familien oder Gattungsname	1	2
Suche3	Eingabefelder semantisch sortieren	2	3
Suche4	Baum nicht aufgeklappt	2	4

Misserfolg heißt in diesem Fall auch, dass die Testpersonen nicht das richtige Abonnement erstellt haben. In Aufgabe 3 war erwartet, dass sie über das Kontextmenü in der Annotationsübersicht (siehe Kapitel 3.3) Annotationen erstellen sollen.

3.6.2 Fazit

Die Änderungen der Beschriftung von „Anzeigename“ in „Abonnementname“ im Abonnement-Editor führte zu einer deutlichen Verbesserung des Verständnisses dieses Feldes unter den Testpersonen.

Tabelle 9: Erfolgreich durchgeführte Aufgaben des dritten Tests

Aufgabe	Erfolg	Teilerfolg	Misserfolg
1 Annotation erstellen	2	1	0
2 Annotation suchen	1	2	0
3 Abonnement konkreter Datensatz	1	0	2
4 Abonnement Familie	3	0	0
5 Abonnement löschen	3	0	0
6 Mail anschauen	-	-	-

3.7 Rückblick auf den Entwicklungsprozess

In diesem Kapitel wurde der Entwicklungsprozess des Benachrichtigungssystems beschrieben. Nutzertests haben sich als gute Methode erwiesen Usability Probleme aufzudecken und Änderungen auf ihre Effektivität zu untersuchen. Der Anteil an erfolgreichen Aufgaben hat sich über den Zeitraum der verschiedenen Tests erhöht. Allerdings zeigen die Ergebnisse auch, dass immer noch Verbesserungspotential besteht. Aufgrund der Vielzahl an Problemen war es auch nicht möglich für alle Lösungen zu entwickeln.
Die Screencasts haben sich als sinnvolle Methode erwiesen, die Tests im Nachhinein noch einmal zu bewerten. Während der Tests war es schwierig Reaktionen der Tester zu beobachten und gleichzeitig Probleme zu dokumentieren. Speziell auf Usability Tests zugeschnittene Software kann dabei helfen Usabilityprobleme effizienter zu dokumentieren.
Im Nachhinein betrachtet, war es nicht notwendig im zweiten Test Aufgaben nicht zu stellen, nur weil die Testperson eine ähnliche Aufgabe nicht erfolgreich bearbeitete. Vielleicht hätte sie bei einem zweiten Versuch mit anderen Vorgaben die korrekte Lösung erarbeitet. Ähnlich war es im dritten Test, bei dem eine Aufgabe gar nicht erst gestellt wurde. Nur weil mit fünf Testpersonen keine Usabilityprobleme aufgedeckt wurden, heißt das nicht, dass keine existieren.

SUS-Auswertung Im zweiten und dritten Test wurde außerdem ein SUS-Fragebogen ausgefüllt. Da die Testpersonen die komplette Anwendung getestet haben und nicht

nur die für diese Arbeit relevanten Teile, konnten keine konkreten Erkenntnisse für die Benachrichtigungen und Abonnements gezogen werden. Im Sinne der Vollständigkeit werden die SUS Ergebnisse in Tabelle 10 dennoch dargestellt. Sie können als Grundlage für zukünftige Usabilitytests verwendet werden.
Nur ein Test hat einen Wert von unter 70 ergeben. Im Vergleich mit den anderen Testwerten, scheint dies ein Ausreißer zu sein und kann auf die eben beschriebene schwierige Sprache der Fragen des SUS-Fragebogen zurückzuführen sein.

	T. 1	T. 2	T. 3	T. 4	T. 5
Test 1	–	–	–	–	
Test 2	72,5	72,5	77,5	75	70
Test 3	55	72,5	70		

Tabelle 10: SUS Auswertung (T: Tester)

Für zukünftige Tests müssen die Fragen eventuell leicht abgeändert werden. Einige Testpersonen haben häufig nachgefragt, was bei den einzelnen Aussagen gemeint sei. Viele störten sich an der Formulierung „unnötig komplex“. Etwas ähnliches schreibt auch Bangor et al. [2]. Sie haben für die Tests auf englisch den Begriff *cumbersome* in *awkward* geändert, weil er für die meisten Personen gebräuchlicher und verständlicher sei.

4 Schluss

In dieser Bachelorarbeit wurde ein Benachrichtigungssystem für die AnnoSys-Webanwendung iterativ mit Hilfe von Nutzertests entwickelt.
Die in Kapitel 2 gestellten Anforderungen konnten, im Rahmen der Möglichkeiten, alle erfüllt werden. Die Nutzer der Webanwendung können Abonnements erstellen und so die Gruppe von Datensätzen einschränken, für die sie Benachrichtigungen erhalten wollen. Zum Zeitpunkt der letzten Tests, waren in der AnnoSys-Anwendung noch keine Rollen für Kuratoren vorhanden. Dadurch war es auch nicht möglich diese Anforderungen konkret zu implementieren.
Durch die vielen Kommentare, die beim lauten Denken der Testpersonen gemacht wurden und die Interviews am Ende der Tests, wurden wichtige Erkenntnisse gesammelt, die zum besseren Verständnis der Zielgruppe beitrugen. So zum Beispiel sind die in der Email dargestellten Informationen für einige Nutzer zu technisch; speziell die Angabe des XML-Selektors. Nicht allen Nutzern ist das Datenformat bekannt, in dem die Datensätze gespeichert werden, geschweige denn sind mit XML vertraut.

Die zuverlässige Integration von Apache Camel und Apache ActiveMQ hat erheblich mehr Zeit in Anspruch genommen als erwartet. Dies war auch ein Grund warum in Kapitel 3.3.1 der Lösungsansatz gewählt wurde, der ohne weitere Anpassung an dem für Camel zuständigen Code auskam. Für den Anwendungsfall dieser Bachelorarbeit – das Versenden von Emails und Nachrichten an einzelne Nutzer – erscheint das Framework zu komplex. Für zukünftige Projekte, bei denen die Anforderung ähnlich stark eingegrenzt sind, sollten eher designierte Frameworks zum Versenden von Emails in Betracht gezogen werden.

Alle Nutzer, die sich zur Zielgruppe der Anwendung zählen, würden die Anwendung erneut benutzen wollen. Dabei ist zu beachten, dass die Nutzer die komplette Anwendung getestet haben und nicht nur die Benachrichtigungen. Ich und das AnnoSys-Team haben festgestellt, wie wertvoll die Rückmeldung der Zielgruppe ist.

4.1 Ausblick

Für viele der in Kapitel 3 aufgeführten Usabilityprobleme wurden noch keine Verbesserungen erarbeiten. Die Ergebnisse dieser Arbeit können dazu genutzt werden um die allgemeine Usability der AnnoSys-Anwendung zu verbessern. So lange noch Usability-Probleme auftreten, sollten auch weiter Nutzertests durchgeführt werden. Einige Probleme traten wiederholt in allen Tests und bei einer Vielzahl der Testpersonen auf. Diese Probleme sollten priorisiert behandelt werden. Zusätzlich können auch automatisierte GUI-Tests eingesetzt werden, um Usabilityproblemen, die auf Programmierfehlern beruhen, vorzubeugen.
Für eine Webanwendung sind neben der Usability auch andere Metriken interessant. So ist zum Beispiel wichtig herauszufinden, wie sich die Anwendung unter hoher Last verhält, wenn viele Nutzer gleichzeitig Annotationen erstellen. Letztendlich wirkt sich das auch wieder auf die Usability der Anwendung aus.

Wie bereits erwähnt, sind die Kuratoren zur Zeit noch nicht in der Anwendung repräsentiert. Es soll aber in Zukunft eine Möglichkeit für Kuratoren geben, Annotationen für ihre Herbarbelege anzuzeigen und zu akzeptieren bzw. abzulehnen. Für diese Funktion wird es sinnvoll sein, ebenfalls früh Nutzertests mit Kuratoren durchzuführen.

5 Anhang

Screenshots vor dem ersten Test

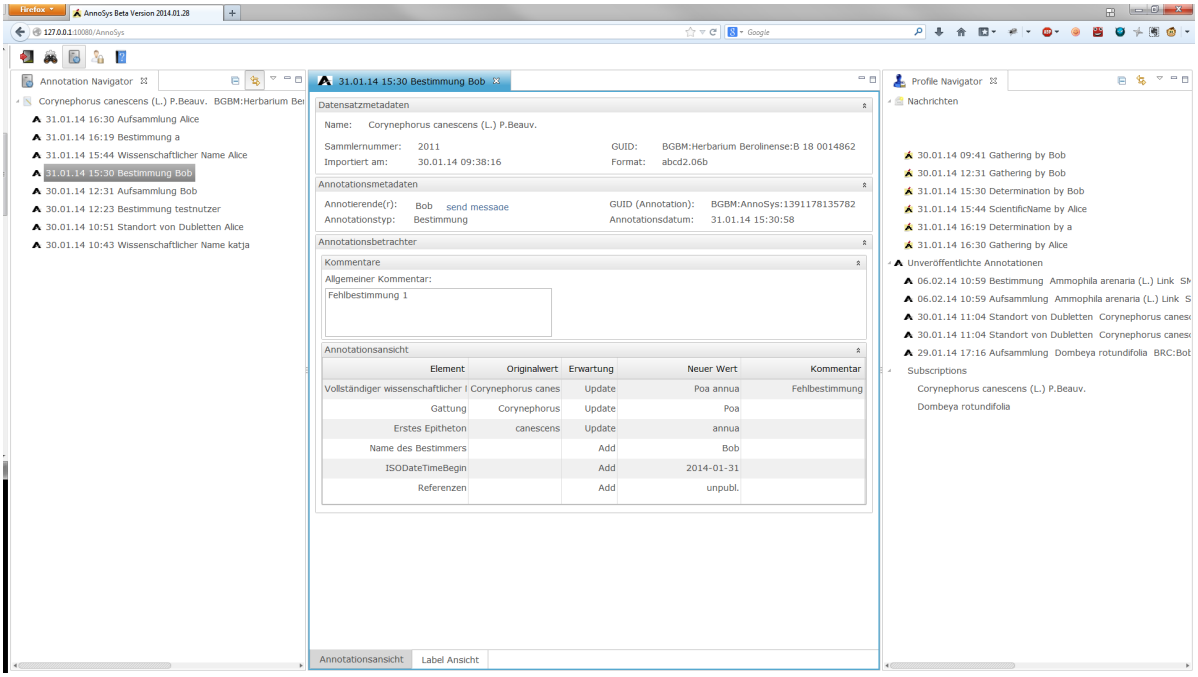


Abbildung 10: Annotation ansehen

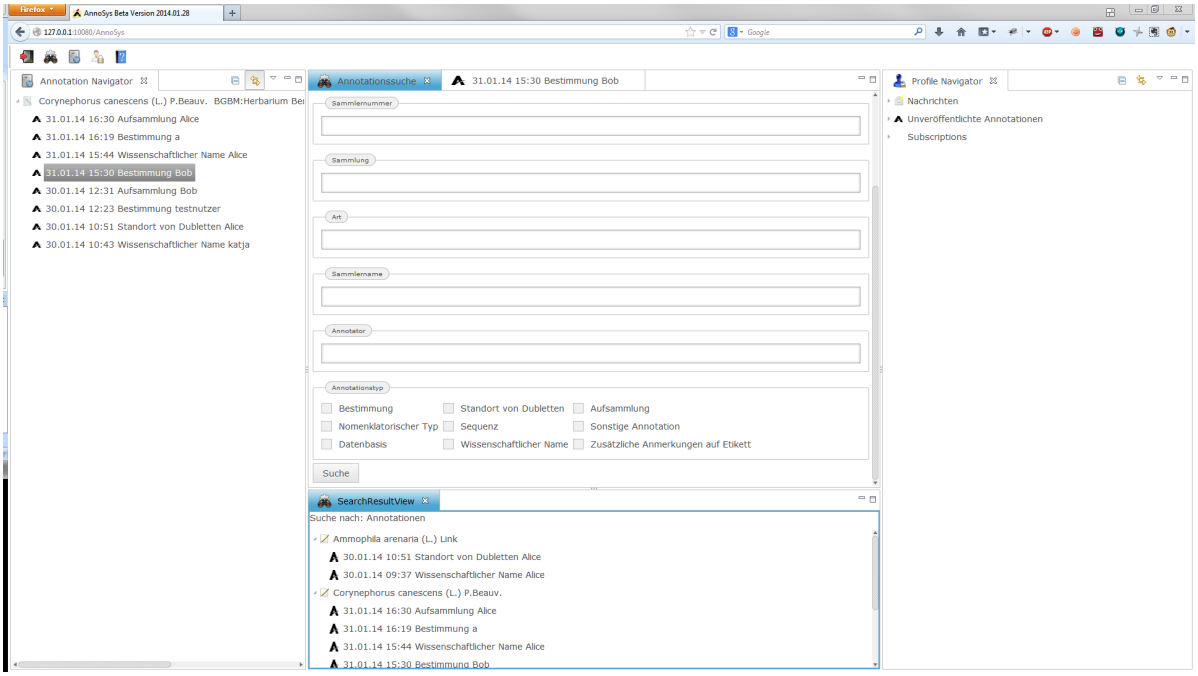


Abbildung 11: Annotationen suchen

Heuristiken nach Nielsen

Sichtbarkeit des Systemstatus	Das System sollte den Nutzer immer darüber informieren, was gerade vorgeht.
Übereinstimmung zwischen System und realer Welt	Das System soll die Sprache des Nutzers sprechen, mit Wörtern, Formulierungen und Konzepten, die dem Nutzer vertraut sind.
Benutzerkontrolle und Freiheit	Nutzer wählen Softwarefunktionen oft versehentlich. Sie brauchen einen klar markierten Notausgang, um einen unabsichtlich erreichten Zustand verlassen zu können, ohne durch einen ausgedehnten Dialog hindurch zu müssen.
Konsistenz und Standards	Die Nutzer sollten nicht überlegen müssen, ob verschiedene Begriffe, Situationen oder Aktionen dasselbe bedeuten.
Fehler vermeiden	Besser noch als gute Fehlermeldungen ist ein sorgfältiges Design, das verhindert, dass überhaupt erst Fehler auftreten.
Erkennen vor erinnern	Mache Objekte, Aktionen und Optionen sichtbar. Der Nutzer sollte nicht gezwungen werden, sich Informationen aus einem Teil eines Dialoges für einen anderen Teil zu merken.
Flexibilität und effiziente Nutzung	Beschleunigungsmöglichkeiten, die der ungeübte Nutzer nicht sieht , können für erfahrene Nutzer die Interaktionsgeschwindigkeit erhöhen, sodass das System sowohl für Neulinge als auch für Experten geeignet ist.
Ästhetisches und minimalistisches Design	Dialoge sollten keine Information enthalten, die irrelevant ist oder selten benötigt wird. Jedes Extra an Information in einem Dialog konkurriert mit relevanten Informationen und vermindert deren relevante Sichtbarkeit.
Unterstützung beim Erkennen, Verstehen und Bearbeiten von Fehlern	Fehlermeldungen sollen in klarer Sprache gegeben werden. Sie sollen das Problem genau beschreiben und konstruktive Lösungen vorschlagen.
Hilfe und Dokumentation	Obwohl es besser ist, wenn das System ohne Dokumentation benutzt werden kann, kann es nötig sein, Hilfe und Informationen mitzugeben. Jede solche Information sollte leicht zu durchsuchen sein, die Aufgabe des Nutzers in den Mittelpunkt stellen und konkrete Schritte zur Ausführung nennen.

System Usability Scale

	Stimme gar nicht zu				Stimme voll zu
Ich kann mir sehr gut vorstellen, das System regelmäßig zu nutzen.					
Ich empfinde das System als unnötig komplex.					
Ich empfinde das System als einfach zu nutzen.					
Ich denke, dass ich technischen Support brauchen würde, um das System zu nutzen.					
Ich finde, dass die verschiedenen Funktionen des Systems gut integriert sind.					
Ich finde, dass es im System zu viele Inkonsistenzen gibt.					
Ich kann mir vorstellen, dass die meisten Leute das System schnell zu beherrschen lernen.					
Ich empfinde die Bedienung als sehr umständlich.					
Ich habe mich bei der Nutzung des Systems sehr sicher gefühlt.					
Ich musste eine Menge Dinge lernen, bevor ich mit dem System arbeiten konnte.					

Szenario

Du bist ein Wissenschaftler im Bereich der Biodiversität. Im Rahmen deiner Arbeit fallen dir diverse Fehler in den Datensätzen verschiedener digitaler Herbarbelege auf. Der allgemeine Arbeitsablauf ist, dass du einen Datensatz im GBIF Portal, oder der BGBM Webseite anschaust und dann annotieren möchtest.

Aufgabe 1 – Annotation erstellen

Öffne den ersten Record auf der Informationsseite. Du glaubst der wissenschaftliche Name der Art sei falsch. Annotiere diesen Eintrag und schlage einen neuen Namen vor.

Aufgabe 2 – Subscription erstellen

In Rahmen deiner Forschung interessierst du dich für alle Annotationen, die für den Gewöhnlichen Strandhafer (Ammophila arenaria) am Botanischen Garten gemacht werden. Melde dich für Annotationen für diese Art in der Sammlung des Botanischen Gartens an. Institut: BGBM, Sammlung: Herbarium Berolinense

Aufgabe 3 – Subscription löschen

Der Gewöhnliche Strandhafer interessiert dich nicht mehr. Lösche die Subscription.

Aufgabe 4 – Annotation finden

Deine Kollegin Alice hat gestern eine Annotation erstellt. Suche nach ihr und öffne die Annotation.

Aufgabe 5 – Nachricht an anderen Nutzer schicken

Du möchtest von Alice wissen, woher sie die Daten für ihre Annotation bekommen hat. Schicke ihr eine Nachricht. Falls du keine Annotation von Alice gefunden hast. Folge dem Link auf der Informationsseite zu Bobs Annotation.

Aufgabe 6 – Mails angucken

Du hast eine Mail bekommen. Schaue dir die Mail an.
Unterbrechung. Alice erstellt wieder eine Annotation [17]

Aufgabe 7 – Messages angucke

Ein neuer Tag, du kommst auf die AnnoSys webseite. Du möchtest wissen ob jemand neue Annotationen erstellt hast

[17] An diesem Punkt habe ich eine Annotation erstellt damit eine Benachrichtigung im Mailfach des Testers landet

Camel Beispiel

Im Folgenden ist ein lauffähiges Codebeispiel gezeigt, wie Nachrichten aus einer ActiveMQ-Queue in eine Datei geschrieben werden. Über das Exchange Objekt kann die Nachricht verändert werden, während sie zwischen zwei Endpoints weitergeleitet wird. Für dieses Beispiel werden die jars camel-core und camel-activemq benötigt.

```
public class CamelTest {

  public static void main(String[] args) throws Exception {
    DefaultCamelContext camelContext = new DefaultCamelContext();
    camelContext.addComponent("activemq", ActiveMQComponent.
        activeMQComponent("vm://localhost?broker.persistent=false"
        ));
    camelContext.addComponent("file", new FileComponent());
    camelContext.start();

    camelContext.addRoutes(new RouteBuilder() {
      @Override
      public void configure() {
        from("activemq:test").process(new Processor() {
              @Override
              public void process(Exchange exchange) throws
                  Exception {
                System.out.println("Nachricht: " +
                                exchange.getIn().getBody());
              }
            })
        .to("file://output_folder");
      }
    });

    Endpoint endpoint = camelContext.getEndpoint("activemq:test")
        ;
    camelContext.createProducerTemplate()
                    .sendBody(endpoint, "Hallo Welt");
    camelContext.stop();
  }
}
```

Literatur

[1] G. Banavar, T. Chandra, R. Strom, and D. Sturman, "A case for message oriented middleware," in *Distributed Computing*. Springer, 1999, pp. 1–17.

[2] A. Bangor, P. T. Kortum, and J. T. Miller, "An empirical evaluation of the system usability scale," *INTL. JOURNAL OF HUMAN–COMPUTER INTERACTION*, vol. 24.

[3] C. M. Barnum, *Usability testing essentials: ready, set... test*. Morgan Kaufmann Publishers Inc., 2010.

[4] W. Berendsohn, "An introduction to the abcd schema v2.0," http://wiki.tdwg.org/twiki/bin/view/ABCD/AbcdIntroduction, June 2007, [Stand: 3. März 2014].

[5] V. Blagoderov, I. Kitching, L. Livermore, T. Simonsen, and V. Smith, "No specimen left behind: industrial scale digitization of natural history collections," *ZooKeys*, vol. 209, no. 0, pp. 133–146, 07 2012. [Online]. Available: http://dx.doi.org/10.3897/zookeys.209.3178

[6] J. Brooke, "SUS – a quick and dirty usability scale," *Usability evaluation in industry*, vol. 189, p. 194, 1996.

[7] I. O. for Standardization, *ISO 9241-11: Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs): Part 11: Guidance on Usability*, 1998.

[8] M. Hertzum and N. E. Jacobsen, "The evaluator effect: A chilling fact about usability evaluation methods," *International Journal of Human-Computer Interaction*, vol. 13, no. 4, pp. 421–443, 2001.

[9] G. Hohpe and B. Woolf, *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Addison-Wesley Professional, 2004.

[10] J. Nielsen, *Usability engineering*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994.

[11] F. Sarodnick and H. Brau, "Methoden der usability evaluation," *Verlag Hans Huber*, 2006.

[12] B. Snyder, D. Bosnanac, and R. Davies, *ActiveMQ in Action*. Manning Publications Co., 2011.

[13] C. Snyder, *Paper prototyping: The fast and easy way to design and refine user interfaces*. Newnes, 2003.

[14] O. Tschöpe, J. A. Macklin, R. A. Morris, L. Suhrbier, and W. G. Berendsohn, "Annotating biodiversity data via the internet," *Taxon*, vol. 62, no. 6, pp. 1248–1258, 2013.