Universität Leipzig
Fakultät für Mathematik und Informatik
Institut für Informatik

Vergleichende Untersuchung zweier Web-Plattformen zur
IT-Unterstützung eines Regionalentwicklungs-Netzwerks

Bachelorarbeit

Leipzig, Oktober 2009

vorgelegt von

Arnold, Patrick
Studiengang Bsc. Informatik

Inhaltsverzeichnis

0 Einleitung

In der vorliegenden Arbeit sollen zwei Web-Plattformen vorgestellt und miteinander verglichen werden, die zur IT-Unterstützung eines Regionalentwicklungsunternehmens verwendet werden können: Cyclos, eine Web-Plattform für Tauschgeschäfte auf Basis von Ersatzwährungen [1] sowie die an der Universität Zürich entwickelte Lernplattform OLAT (Online Learning and Training) [2]. Beide Plattformen sind weitgehend in Java geschrieben und stehen unter GPL-Lizenz. Sie besitzen einen beachtlichen Funktionsumfang und können in einem umfangreichen Kontext eingesetzt werden. Während Cyclos direkt für regionale Entwicklungsgesellschaften konzipiert ist, ist OLAT eine reine Lernplattform, die vor allem an Schulen, Hochschulen und Universitäten zum Einsatz kommt. Daher wird ein wesentlicher Aspekt der Arbeit sein, die Erweiterungsmöglichkeiten von OLAT bzw. die funktionalen Voraussetzungen zu untersuchen, damit das System bei entsprechender Anpassung auch in o.g. Netzwerken eingesetzt werden kann.

Im ersten Kapitel der Arbeit werden regionale Entwicklungsgesellschaften näher erläutert sowie deren Funktionsprinzip dargelegt. Außerdem werden hierbei auch die Anforderungen an ein Softwaresystem skizziert, welches in einem solchen Unternehmen verwendet werden soll. Dazu gehören nicht nur der Funktionsumfang, sondern auch das Einsatzgebiet, die Zielgruppe sowie die konkreten Leistungs- und Sicherheitsanforderungen. Als Beispiel soll hierbei die Anhalt Dessau AG im Mittelpunkt stehen, obgleich auch noch auf einige weitere Unternehmen aus dieser Branche eingegangen wird, um das Konzept solcher Entwicklungsgesellschaften besser zu veranschaulichen. Am Ende dieses Kapitels werden dann die genauen Anforderungen feststehen, die ein Softwaresystem erfüllen muss, welches in regionalen Entwicklungsgesellschaften eingesetzt werden soll.

Im Anschluss an das erste Kapitel werden die zwei Softwareprojekte OLAT und Cyclos genauer untersucht werden. Ein wesentliches Ziel ist dabei, zu untersuchen, inwiefern das jeweilige System die zuvor ermittelten Anforderungen erfüllt. Das setzt natürlich eine genaue Analyse des Funktionsumfangs der jeweiligen Software voraus, aber auch die Untersuchung der Erweiterungsmöglichkeiten sowie der Programmarchitektur. Im Kapitel 2 wird dabei zunächst allgemein erläutert, wie Web-Plattformen wie Cyclos und OLAT aufgebaut sind, bevor dann im 3. Kapitel das Cyclos Projekt, und im 4. Kapitel das OLAT genauer untersucht werden.

Im 5. Kapitel wird schließlich ein Vergleich beider Plattformen vorgenommen. Ziel ist es dabei, die Vor- und Nachteile der einzelnen Softwaresysteme aufzuzeigen, ihr zu empfehlendes Einsatzgebiet sowie den Grad der Übereinstimmung mit den im ersten Kapitel aufgezeigten Anforderungen. Eine Zusammenfassung rundet die Arbeit schließlich ab.

1 Regionalentwicklungs-Netzwerke

1.1 Einleitung

Regionalentwicklungs-Netzwerke sind Unternehmen, welche das Ziel verfolgen, mit geeigneten Konzepten und Strategien die Wirtschaft innerhalb einer Region zu stärken. Sie besitzen dabei einen bestimmten regionalen Einflussbereich und versuchen sowohl Unternehmen als auch Privatpersonen auf der Grundlage von Kommunikation und Kooperation zusammenzuführen und damit einen regionalen Wirtschaftskreislauf zu schaffen. Ein solcher Wirtschaftskreislauf wird natürlich niemals völlig unabhängig von der Wirtschaft eines Landes bzw. der Weltwirtschaft sein. Vielmehr ist er eingebettet in überregionale Wirtschaftskreisläufe. So kann z.B. innerhalb des Landes Mecklenburg-Vorpommern durchaus ein regionaler Wirtschaftskreislauf entstehen, in dem regionale Güter (z.B. Fisch, Getreideprodukte etc.) sowie verschiedene Dienstleistungen (z.B. Handwerks- und Bauarbeiten) gehandelt bzw. durchgeführt werden. An dem regionalen Wirtschaftskreislauf erfolgreich teilnehmende Unternehmen setzen dabei einen Großteil ihrer Erzeugnisse innerhalb der Region ab und werden zum Herstellen der Erzeugnisse auch auf Ressourcen aus der Region zurückgreifen. Ein Landwirtschaftsunternehmen wird somit seine Agrarprodukte weitgehend in der Region verkaufen und, soweit möglich, Betriebsmittel (z.B. Fahrzeuge, Dünger etc.) und Rohstoffe (z.B. Wasser, Getreidesamen) aus der Region beziehen. Bei genügend Unternehmen, die auf ähnliche Weise agieren, entsteht somit ein regionales Netzwerk.

Einige Güter und Dienstleistungen können jedoch in einer Region nicht produziert oder abgesetzt werden, da sie von anderen Faktoren abhängig sind, welche die Region nicht bieten kann. Ein regionaler Wirtschaftskreislauf steht somit im Austausch mit anderen bzw. mit größeren Wirtschaftskreisläufen. So werden aus Mecklenburg-Vorpommern bspw. Güter nach Baden-Württemberg exportiert (z.B. Fisch), und ebenso wird bspw. Wein von Baden-Württemberg nach Mecklenburg-Vorpommern importiert. Die Abhängigkeit besteht, da Baden-Württemberg auf Grund seiner Lage keinen Fischfang betreiben kann; Mecklenburg kann auf Grund klimatischer Bedingungen keinen Wein anbauen. Somit entsteht ein landesweiter Wirtschaftskreislauf, in dem viele regionale Wirtschaftskreisläufe eingebettet sein können. Aber auch der deutsche Wirtschaftskreislauf ist keineswegs isoliert, sondern wiederum in die Weltwirtschaft eingebettet. So werden deutsche Unternehmen Rohstoffe importieren, die sie vor Ort nicht gewinnen können und Waren exportieren, die andere Länder womöglich nicht fertigen können.

Vor allem die Bereiche Handwerk, Landwirtschaft, Lebensmittelindustrie, Baugewerbe und Bildung sind jedoch regional relativ unabhängig. Diese können somit die Grundlage für einen regionalen Wirtschaftskreislauf bilden, in dem es in dieser

Arbeit gehen soll. Leider entwickeln sich regionale Wirtschaftskreisläufe bisweilen nicht besonders gut, da aus unterschiedlichen Gründen die günstigen Voraussetzungen oft nicht genutzt werden. So kommt es z.B. häufig vor, dass für den Bau einer Kreisstraße eine Baufirma aus einem anderen Bundesland bzw. einer anderen Region beauftragt wird. Naheliegend wäre es jedoch, eine Firma aus dem entsprechenden Landkreis mit dem Bau der Kreisstraße zu beauftragen. Nur auf diese Weise könnte ein regionalwirtschaftlicher Kreislauf entstehen, in dem die Region der alleinige Profitierende der wirtschaftlichen Transaktionen ist und geschaffenes Geld nicht in andere Regionen abfließt. In den letzten Jahren konnte man bei der Bevölkerung jedoch einen leichten Trend zur Regionalität beobachten. So werden regionale Produkte (insbesondere Lebensmittel) gegenüber anderen Produkten immer häufiger vorgezogen. Dies ist eine gute Voraussetzung für die Entstehung regionaler Wirtschaftskreisläufe.

Regionalentwicklungs-Netzwerke versuchen nun, regionale Wirtschaftskreisläufe aufzubauen und aufrechtzuerhalten, was in erster Linie auf der Basis von Tauschgeschäften realisiert wird. An den Entwicklungsgesellschaften beteiligen sich dabei regionale Unternehmen oder Privatpersonen, die miteinander in Beziehung treten und Dienstleistungen oder Güter tauschen. Damit die Tauschgeschäfte vergleichbar werden, bieten solche Netzwerke meist eine oder mehrere Ersatzwährungen an. Zudem werden oft Anreize geboten, um Unternehmen und Privatpersonen zur Beteiligung an dem Konzept zu ermuntern. Auf Grund der regionalen Tauschgeschäfte und auf Grund der Tatsache, dass die Ersatzwährungen nur innerhalb der Region und i.A. nur von den an diesem Konzept teilhabenden Unternehmen und Personen akzeptiert werden, soll so die Wirtschaft einer bestimmten Region gestärkt und ein regionaler Wirtschaftskreislauf gegründet werden. Neben der Stärkung der regionalen Wirtschaft ergeben sich dabei auch noch zahlreiche weitere Vorteile für teilhabende Unternehmen und Personen, die später noch vorgestellt werden.

Wie bereits erwähnt, bieten Regionalentwicklungs-Netzwerke i.A. mindestens eine Ersatzwährung, die meist durch Umtausch einer offiziellen Währung (z.B. Euro) erworben werden kann. Ein Zurücktauschen ist meist nicht möglich, damit das einmal umgetauschte Geld in der Region verwendet wird und somit ausschließlich dieser Region zu Gute kommt. Mittels der Ersatzwährung können dann Güter und Leistungen von Unternehmen bezahlt werden, die sich an dem Konzept beteiligen. Hierbei sind oft auch Split-Beträge möglich (z.B. 50 % in Euro und 50 % in der entsprechenden Ersatzwährung), auf die sich Auftraggeber und Auftragnehmer einigen können. Damit zeigt sich, dass man Beträge in der Ersatzwährung nicht nur durch Umtauschen erhalten kann, sondern auch durch das Erbringen einer Leistung bzw. das Verkaufen einer Ware (also durch sog. Tauschgeschäfte).

Das Grundprinzip bei Tauschgeschäften ist stets gleich. Ein Unternehmen oder eine Person A erweist einem Unternehmen oder einer Person B eine bestimmte Dienstleistung (oder liefert ein bestimmtes Gut) und erhält dabei von B eine Bezahlung in der entsprechenden Ersatzwährung. B hat nun womöglich einen negativen Kontostand. Irgendwann leistet B aber einem Unternehmen bzw. einer Person C eine Dienstleistung (oder liefert ein bestimmtes Gut) und gleicht damit seinen Kontostand wieder aus. B konnte also eine Leistung oder ein Gut erhalten, ohne dafür

finanzielle Mittel aufgebracht zu haben. Der Vorteil ist also vor allem, dass Unternehmen und Privatpersonen Leistungen oder Güter erhalten können, auch wenn sie momentan kein Geld zur Verfügung haben. Sie bezahlen mit ihren eigenen Produkten bzw. ihren Dienstleistungen; Privatpersonen bezahlen mit ihrem „Talent“. Eine Person A kann z.B. drei Stunden die Kinder der Nachbarin B beaufsichtigen, und im Gegensatz dazu kann B dann z.B. A bei Gartenarbeiten helfen. Auch im privaten Bereich funktionieren solche Tauschgeschäfte also bestens.

Obwohl Tauschgeschäfte meist die Grundlage eines Regionalentwicklungsunternehmens bilden, sind solche Geschäfte nicht unbedingt notwendig, um die regionale Wirtschaft zu fördern. So wäre z.B. auch eine einfache Plattform ausreichend, auf der sich Anbieter und Nachfrager einer Region treffen und miteinander ins Geschäft kommen. Das eigentliche Geschäft kann dann auch ebenso außerhalb des Systems durchgeführt werden, also völlig unabhängig von dem Regionalentwicklungsunternehmen. Auch ohne Tauschhandel oder Ersatzwährungen hat dieses Unternehmen dann einen positiven Effekt auf die regionale Wirtschaft erzielt, da durch dessen Wirken regionale Unternehmen geschäftliche Beziehungen eingegangen sind.

Aus dem Konzept, das regionale Entwicklungsgesellschaften verfolgen, ergeben sich eine Reihe von Vorteilen, wie sie bspw. auch vom Regiogeld e.V.[1] und der Anhalt Dessau AG[2] beschrieben werden. Einige bedeutsame Vorzüge sind dabei:

- Stärkung der Region, da die Gewerbesteuer in der Region bezahlt wird.
- Stärkung der regionalen Unternehmen.
- Schaffung oder zumindest Erhalt von Arbeitsplätzen in der Region.
- Durch das Regionalentwicklungs-Netzwerk lernen sich Unternehmen kennen und es können neue Geschäftsbeziehungen entstehen.
- Unternehmen können Leistungen und Güter erhalten, auch wenn sie finanziell dazu nicht in der Lage sind (sie bezahlen mit ihren eigenen Leistungen oder Gütern).
- Unternehmen können Waren schneller absetzen.
- Die Bezahlung von Gütern und Dienstleistungen findet i.A. unverzögert statt.
- Teilweise Einsparung von Aufwand und Formalitäten.

Natürlich können hierbei auch einige Nachteile aufgezeigt werden. Ein Nachteil ist z.B., dass das erworbene „Ersatzgeld“ nur gegen andere Leistungen und Güter der an dem Konzept teilhabenden Unternehmen eingetauscht werden kann, was bei Ersatzwährungen also zu einer deutlichen Einschränkung an Flexibilität führt. Auch sind diese Währungen i.A. nicht gedeckt, zumindest nicht durch den Staat. Stellt sich bspw. heraus, dass das Regionalentwicklungsunternehmen verschuldet ist und Insolvenz anmelden muss, so wäre es möglich, dass das Guthaben teilhabender Personen und Unternehmen vernichtet und kein Schadensersatz gewährt wird. Ein solches Risiko besteht für offizielle Landeswährungen meist nicht.

Trotzdem erscheinen die Chancen und Vorteile höher als die möglichen Nachteile.

[1] http://www.regiogeld.de/regiogeld.html
[2] http://www.dessau-ag.de/Bereich_Barter/vorteile.php

Eine wichtige Voraussetzung ist jedoch, dass genügend Unternehmen bzw. Privatpersonen an dem Konzept aktiv teilhaben; sonst können sich in der Tat deutliche Nachteile für die beteiligten Unternehmen ergeben.

1.2 Unternehmen

1.2.1 Anhalt Dessau AG

Die Anhalt Dessau AG ist ein Beispielvertreter eines regionalen Entwicklungsunternehmens und soll nun näher vorgestellt werden. Das Unternehmen mit Sitz in Dessau, Sachsen-Anhalt, bietet seit Oktober 2005 wirtschaftliche Transaktionen auf Basis von Tauschgeschäften an.[3] Dabei werden Unternehmen und Privatpersonen separat behandelt.[4] Unternehmen gehören dem sog. *Mitteldeutschen Barterring* an, in dem die Währung *Barter* als Verrechnungseinheit verwendet wird. Hierbei können Unternehmen und Freiberufler beliebig Waren und Dienstleistungen handeln. Die Bezahlung kann in Barter, in Euro oder im Splitverhältnis erfolgen. Sie muss nach spätestens 6 Monaten erfolgen, sonst wird das Unternehmen zur Barzahlung der Leistung bzw. Ware aufgefordert.[5]

Analog dazu ist der *Dessauer Tauschring*, bei dem die Mitglieder Privatpersonen sind und keine gewerblichen Tätigkeiten ausüben. Dabei dienen sog. *Talente* als Bezahlungseinheit. Ein Talent entspricht einer Arbeitsleistung von 10 Minuten.[6]

Der Wechselkurs zwischen Euro, Bartern und Talenten ist recht einfach gehalten: Ein Barter entspricht einem Talent und ein Talent entspricht einem Euro. Somit können Dienstleistungen und Waren angemessen bewertet werden. Barter und Talente können auch gekauft werden, ein Rücktausch in Euro ist jedoch grundsätzlich nicht möglich.[7]

Als eine dritte Währung werden von der Anhalt Dessau AG unter dem Slogan „Initiative Dessau - Arbeit für Anhalt e.V.“ sog. *DeMark* (Dessauer Mark, kurz DeM) angeboten [3]. Diese haben ebenfalls einen Wechselkurs von 1:1 in Bezug auf den Euro und können in der Wechselstube der Anhalt Dessau AG erworben werden. Als Anreiz werden für 10 eingetauschte Euro 11 DeMark ausgegeben, mit denen dann in derzeit etwa 46 regionalen Geschäften und Unternehmen („Akzeptanzstellen“) bezahlt werden kann.[8] Dabei haben die gekauften DeMark-Scheine lediglich

[3] http://www.dessau-ag.de/history.php
[4] http://www.dessau-ag.de/presse/Broschuere_Buergersolar.pdf
[5] http://www.dessau-ag.de/Bereich_Barter/faq.php
[6] http://www.dessau-ag.de/presse/Nutzungsrichtlinien_DeM.pdf
[7] http://www.dessau-ag.de/Bereich_Regio/richtlinien.php
[8] http://www.dessau-ag.de/Bereich_Regio/regiomitgl.php

eine Gültigkeit von 4 Monaten, danach findet ein Werteverfall von 10 % pro Monat statt. Somit soll das Mitglied dazu animiert werden, das erworbene Geld möglichst schnell wieder ausgeben.[9]

Damit die Transaktionen stattfinden können, verwendet das Unternehmen eine eigens errichtete Kooperationsplattform, die weitgehend in PHP geschrieben ist. Auf dieser Plattform können sich Mitglieder einloggen, Angebote erstellen und suchen, sowie mit Anbietern in Kontakt treten. Bei weiterem Interesse können sie einen Vertrag abschließen und nach Ausführung der Leistung den Bezahlvorgang durchführen (Transaktion).

Eine Besonderheit der Anhalt Dessau AG ist der Einsatz von Kundenbetreuern. Jedes Mitglied ist dabei einem Kundenbetreuer zugeordnet, der als Ansprechpartner, Vermittler und Betreuer dient. Registriert sich ein neues Mitglied, so wird dieses zunächst als *Kontakt* betrachtet und einem Kundenbetreuer zugeordnet; es ist noch kein vollständiges Mitglied. Der Kundenbetreuer kann den Kontakt später in einen Barterkunden umwandeln, so dass das Mitglied ein Konto erhält und nun auch Angebote erstellen und wahrnehmen kann.

Obwohl Tauschgeschäfte und Regionalgeld zum grundlegenden Funktionsumfang der Anhalt Dessau AG gehören, gibt es noch weitere kooperative Funktionalitäten, die das Unternehmen in Zukunft umsetzen möchten. Im Oktober 2008 wurden von der Unternehmensleitung u.a. folgende Ideen vorgeschlagen, die dem regionalen Entwicklungskonzept hinzugefügt werden können:

- **„Mitfahrgelegenheit für Güter“** – Diesem Konzept liegt folgende Idee zu Grunde: Jeden Tag legen viele Personen mit ihrem privaten Fahrzeug größere Strecken zurück, z.B. von einer Stadt A nach einer Stadt B. Dabei wird oft viel Potenzial verschenkt, da sie auf diesen Strecken theoretisch auch weitere Personen sowie Güter mit an den Zielort befördern könnten. Die Idee, Personen auf solchen Fahrten mit zu befördern, gibt es längst; Fahrgemeinschaften werden heute bereits reichhaltig angeboten, wobei www.online-mitfahrzentrale.com z.B. ein bedeutsamer Vertreter ist. Für Güter gibt es ein solches Konzept hingegen i.A. nicht, das heißt, große Güter müssen per Paketdienst versendet werden, was bisweilen aufwendig und kostenintensiv ist. Die Anhalt Dessau AG überlegt daher, mit einer „Mitfahrgelegenheit für Güter“ auch die Möglichkeit zu schaffen, dass Fahrer die Güter anderer Personen mit zu dem Zielort befördern.
- **Fahrrad-Plattform** – Mitglieder sollen online Fahrräder mieten können, wobei Start- und Zielort nicht zwangsweise gleich sein müssen.
- **Car-Sharing** – Mitglieder sollen die Möglichkeit haben, Autos von anderen Mitgliedern zu leihen, wobei Zeitraum und Konditionen online vereinbart werden. Außerdem soll ein Bewertungssystem zur Verfügung stehen, so dass Mitglieder bewertet werden können. Jemand, der bereit wäre, sein Auto zu verleihen, soll somit einschätzen können, ob ein Interessent als zuverlässig gilt oder nicht.
- **Regio-Card** – Statt der DeMark-Scheine soll eine „Regio-Card“, also eine Art Kreditkarte, verwendet werden, mit der DeMark-Beträge ausgezahlt werden

[9] http://www.ini-dessau.de/DeM.php

können. Hierfür wird zum einen eine Chipkarte benötigt und zum anderen ein Kassensystem mit entsprechender Kassen-Software, so dass die Bezahlung in den Akzeptanzstellen ermöglicht werden kann.

Dabei ist dies im Grunde nur ein Auszug der Ideen, die damals von der Anhalt Dessau AG vorgestellt wurden.[10] Das Unternehmen hat noch andere Funktionalitäten in Erwägung gezogen; wann und in welchem Maße diese Funktionen umgesetzt werden können, ist jedoch sehr unsicher, da es u.a. auch von der finanziellen Situation der Anhalt Dessau AG sowie von möglichen Investoren bzw. staatlichen Fördermitteln abhängt.

Auch wenn die o.g. Funktionalitäten bisweilen deutlich von den sog. Tauschgeschäften bzw. dem Regionalgeld-Konzept abweichen, ist noch immer der kooperative Charakter zu erkennen. Genau das macht ein regionales Entwicklungsnetzwerk aus. Es ist nicht zwangsweise auf den Tausch von Gütern und Dienstleistungen beschränkt, es umfasst im Grunde alle wirtschaftlichen Transaktionen, die zwischen Personen innerhalb einer bestimmten Region durchgeführt werden können. Kooperation ist ein ganz wesentliches Merkmal eines solchen Netzwerks und wird in der Anforderungsanalyse noch näher erläutert.

Im Nachfolgenden sollen aber erst einmal einige weitere Unternehmen vorgestellt werden, die ebenfalls als regionale Entwicklungsgesellschaften betrachtet werden können.

1.2.2 Weitere Beispiele

Die Genossenschaft regioStar e.G. mit Sitz im Berchtesgadener Land bietet ebenfalls eine Ersatzwährung zur regionalen Förderung der Wirtschaft an [4]. Hierbei können Unternehmen und Privatpersonen so genannte *Sterntaler* beziehen, wobei ein Sterntaler dem Wert eines Euros entspricht. Verschiedene Unternehmen bieten bereits die Bezahlung in Sterntaler an; die Gesellschaft verspricht zudem, dass von jedem überwiesenen Betrag 3 % in Förderprojekte investiert werden.[11]

Ein weiteres System, das bereits seit Oktober 2004 auf diesem Markt agiert, ist die UrstromTaler e.G. Hierbei handelt es sich um eine Regionalwährung für das Land Sachsen-Anhalt, deren Funktionsprinzip in etwa dem der Anhalt Dessau AG entspricht [5]. Bei dem Unternehmen können Euros in sog. *Urstromtaler* im Verhältnis 1:1 umgetauscht und auf einem Konto verwaltet werden. Akzeptanzpartner nehmen bei der Bezahlung von Waren und Leistungen 10 bis 100 % des Urstromtalers an und zahlen ggf. selbst in Urstromtalern.[12] Wie bei der Anhalt Dessau AG ist das Umtauschen von Guthaben und Wertgutscheinen nicht möglich und ebenso hat der

[10] Die Anhalt Dessau AG hat im Oktober 2008 im Rahmen eines Regionalweb-Projekts am Lehrstuhl für Betriebliche Informationssysteme der Universität Leipzig u.a. die o.g. Ideen vorgestellt.
[11] http://www.regiostar.com/15.0.html
[12] http://urstromtaler.de/archiv/33

Urstromtaler nach 6 Monaten einen Werteverfall von 5 %.[13] Auch in diesem System ist also das vordergründige Ziel, das Geld möglichst schnell wieder in der Region auszugeben und somit den Umlauf zu fördern.[14]

Außerhalb von Deutschland gibt es ebenfalls eine Vielzahl solcher bzw. ähnlicher Unternehmen. Im internationalen Sprachraum hat sich dabei der Begriff LETS (Local Exchange Trading System) durchgesetzt, womit lokale Netzwerke bezeichnet werden, in denen Güter und Leistungen ohne reales Geld gehandelt bzw. getauscht werden können [6]. Vor allem in Großbritannien haben sich dabei bereits viele solcher Netzwerke etabliert. North London LETS ist beispielsweise ein solches Netzwerk [7]. Dieses unterscheidet sich jedoch deutlich von den bisher vorgestellten Netzen. So werden bei North London LETS nur allgemeine Dienstleistungen (Baby-Sitting, Gartenarbeiten, Haushaltshilfen etc.) angeboten. Zudem finden hierbei keinerlei finanzielle Transaktionen statt. Für eine Dienstleistung wird ausschließlich in sog. *Pledges* bezahlt, die aber nicht in Pfund umgetauscht, und ebenso wenig mit Pfund bezahlt werden können. Damit dieses System funktionieren kann, sind negative Kontostände somit grundsätzlich erlaubt.

Auch bei North London LETS steht der Vorteil im Vordergrund, dass man notwendige Dienstleistungen erhalten kann, auch wenn man hierfür kein Geld zur Verfügung hat. Die Förderung der Region, insbesondere aus wirtschaftlicher Sicht, ist bei diesem Konzept jedoch eher gering. In dieser Beziehung unterscheidet sich North London LETS deutlich von der Anhalt Dessau AG bzw. den beiden anderen vorgestellten Gesellschaften.

Der Talente-Tausch-Graz in Österreich ist eine regionale Entwicklungsgesellschaft, bei der die Ersatzwährung Arbeitsstunden entspricht [8] [9]. Das heißt, jedes Mitglied verfügt über ein Guthaben, das angibt, wie viele Stunden es für andere Unternehmen gearbeitet hat. Dieses Guthaben darf jedoch ein Minus von 20 Stunden und ein Plus von 50 Stunden nicht überschreiten bzw. unterschreiten. Leistungen werden somit in Arbeitsstunden bezahlt. Zur Finanzierung des Vereins müssen Mitglieder dabei 12 Euro im Jahr bezahlen und es werden für den organisatorischen Aufwand 3 Arbeitsstunden pro Jahr vom Konto gestrichen. Der Betreiber beschreibt das Ziel des Netzwerks als „die Nutzung brachliegender Fähigkeiten (TALENTE) unter Freunden und Nachbarn und regionaler Ressourcen unter Berücksichtigung eines harmonischen Verhältnisses einerseits zwischen Menschen untereinander und Mensch und Natur andererseits.“ [9, S. 1]. Er betont dabei, dass keine Gewinnerzielungsabsicht und gewerbliche Nutzung ausgeübt werden darf.

[13] http://urstromtaler.de/archiv/112
[14] http://urstromtaler.de/archiv/category/fuer-sich-werben

1.3 Geschäftsvorfälle

1.3.1 Überblick

Um im nächsten Abschnitt die genauen Anforderungen an ein Softwaresystem zur Unterstützung regionaler Entwicklungsgesellschaften zusammenstellen zu können, sollen in diesem Abschnitt zunächst konkrete Geschäftsvorfälle dargelegt werden, die sich in einem solchen Unternehmen vollziehen.

Es gibt im wesentlichen drei grundlegende Geschäftsvorfälle:

1. **Angebot anbieten:** Angebot erstellen – Angebot bearbeiten – Kommunikation mit einem Interessenten – Abschließen eines Vertrags – Angebot löschen – Angebot ausführen (reale Welt) – Erhalt der Vergütung.
2. **Angebot annehmen:** Angebot suchen – Informationen über den Anbieter einsehen – Kommunikation mit dem Anbieter – Abschließen eines Vertrags – Güter/Dienstleistung erhalten (reale Welt) – Überweisung der Vergütung.
3. **Neues Mitglied registrieren:** Registrierung als Kontakt – Zuweisung eines Kundenbetreuers – Registrierung als neues Mitglied.

Der Fall 3 ist jedoch nicht allgemeingültig, sondern ein spezieller Geschäftsvorfall der Anhalt Dessau AG. Nichtsdestoweniger muss ein Softwaresystem auch diesen Fall berücksichtigen.

Bei Kooperationsplattformen ist grundsätzlich zwischen einem Angebot und einem Gesuch zu unterscheiden. Bei einem Angebot bietet der Angebotsersteller A (Anbieter, Auftragnehmer) ein Gut bzw. eine bestimmte Leistung an, die ein Mitglied B (Kunde, Käufer, Auftraggeber) annimmt und A dafür bezahlt. Bei einem Gesuch handelt es sich bei dem Ersteller um einen Kunden, der einen Anbieter sucht, welcher seine Bedürfnisse deckt.

In der realen Wirtschaft sind Angebote eher üblich als Gesuche. Kunden haben Bedürfnisse und suchen nach einem Anbieter, der ihre Bedürfnisse erfüllen kann. Es ist eher unüblich, dass Kunden eine Art Ausschreiben bzw. Gesuch machen und darauf warten, dass ein Unternehmen bzw. eine Firma auf sie zukommt. Aus diesem Grund soll der Fokus in dieser Arbeit auf Angeboten liegen, und nicht auf Gesuchen (unter dem Begriff *Angebot* soll zukünftig sowohl ein Angebot als auch ein Gesuch zu verstehen sein). Dennoch können natürlich auch Gesuche in einer Kooperationsplattform erstellt werden. Typische Beispiele hierfür sind vor allem Nachhilfe, Babysitten und ähnliche Dienstleistungen.

Im Übrigen ändern sich bei Gesuchen die beiden o.g. Geschäftsprozesse nur geringfügig. Es ändert sich lediglich die Bezahlrichtung, das heißt, bei einem Angebot ist der Ersteller der Akteur und erhält von einem Kunden eine Vergütung, bei einem Gesucht ist der Ersteller hingegen derjenige, der Waren bzw. eine Dienstleistung erhält und somit eine Vergütung zahlen muss.

1.3.2 Fall 1: Angebot anbieten

Beim Anbieten eines Angebots wird der Anbieter zunächst ein neues Angebot erstellen. Hierbei wird er dem Angebot mindestens einen Titel geben und eine Beschreibung hinzufügen, die Rahmenbedingungen, technische Informationen etc. enthält. Die meisten Anbieter werden zudem mindestens ein Bild zu dem Angebot hochladen, besonders anspruchsvolle Mitglieder werden vielleicht sogar einen kurzen Videoclip zum Angebot hinzufügen wollen, um bspw. die Funktionsweise eines speziellen Geräts zu demonstrieren etc.

Der Anbieter hat zudem immer die Wahl, einen festen Preis anzugeben oder ihn offen zu halten, um mit einem Interessenten ggf. einen angemessenen Preis aushandeln zu können. Die Bezahlung kann in der Ersatzwährung erfolgen oder im Split-Verhältnis mit anderen (Ersatz-) Währungen. Auch muss berücksichtigt werden, dass verschiedene Güter und Dienstleistungen zeitlichen Einschränkungen unterliegen. So müssen Lebensmittel innerhalb eines bestimmten Zeitraums abverkauft werden, da sie ein bestimmtes Haltbarkeitsdatum besitzen. Verschiedene Bauarbeiten können nicht im Winter erledigt werden, müssen also noch vor Winterbeginn durchgeführt werden. Bootstouren können nur durchgeführt werden, wenn der Wasserpegel es zulässt. Somit wird der Anbieter sein Angebot also unter Umständen nur in einem bestimmten Zeitraum bzw. bis zu einem bestimmten Stichtag anbieten.

Ist das Angebot erstellt, so muss es natürlich für alle anderen Mitglieder im System sichtbar werden. Der Anbieter wird ggf. im Nachhinein einige Punkte des Angebots ändern. Das kann die Beschreibung des Angebots umfassen, aber auch den Preis oder den Zeitraum, in dem es gilt. Der Anbieter wird ggf. weitere Bilder hineinstellen oder bereits hochgeladene Bilder wieder herausnehmen. Ein Angebot kann also nach dem Erstellen beliebig geändert werden.

Zeigt jemand an dem Angebot Interesse, so werden sicher beide Mitglieder den Wunsch verfolgen, Einzelheiten zu diskutieren. Der Anbieter wird z.B. einen Preis aushandeln bzw. den zuvor festgelegten Preis ggf. ändern. Er wird möglicherweise auch die Konditionen ändern oder dem Interessenten weiterführende Informationen geben bzw. dessen Fragen beantworten. Dies alles geschieht natürlich diskret; andere Mitglieder wissen nicht, wie viele weitere Interessenten es zu einem Angebot gibt und was diese mit dem Anbieter besprochen haben.

Sind sich Anbieter und Nachfrager einig, so muss ein revisionssicherer Vertrag ausgearbeitet werden. Dieser sollte nun einen eindeutigen, festen Preis und die konkreten Rahmenbedingungen und Konditionen enthalten. Auch kann ein Datum festgelegt werden, bis wann das Angebot auszuführen ist. Zudem muss der Anbieter festlegen können, bis wann die Bezahlung zu erfolgen hat (z.B. im Voraus bzw. bis zu einem bestimmten Stichtag, in Raten oder einmalig etc.).

Ist der Vertrag abgeschlossen, so wird der Anbieter das Angebot ggf. löschen, wobei er das Angebot ebenso gut auch zu jedem anderen Zeitpunkt löschen können muss. Hierbei gibt es sicher auch Angebote, die permanent bestehen (z.B. Stadtführungen, Verkauf von Erzeugnissen etc.). Solche Angebote können also nach Vertragsabschluss

bzw. Ausführung bestehen bleiben; das Abschließen eines Vertrags impliziert somit nicht, dass das Angebot entfernt wird.

Die Ausführung des Angebots, also entweder die Lieferung eines Gutes oder das Erbringen einer Dienstleistung, geschieht außerhalb der Kooperationsplattform, also in der realen Welt. Üblicherweise wird der Empfänger der Ware bzw. Dienstleistung dem Anbieter den Empfang quittieren, so dass auch in der realen Welt ein Nachweis über die ausgeführte Dienstleistung bzw. die versendete Ware besteht.

Spätestens nachdem die Ware ausgeliefert wurde, muss der Auftraggeber den Auftragnehmer die entsprechende Vergütung bezahlen. Hierzu wird er einen bestimmten Betrag überweisen und wie bei einer Buchung im Bankwesen eine bestimmte Beschreibung bezüglich der Überweisung angeben. Beträge in offizieller Währung (z.B. Euro) werden jedoch nicht über das System abgewickelt, sondern finden in der realen Welt statt (z.B. Überweisung auf ein Bankkonto).

1.3.3 Fall 2: Angebot annehmen

Ein potentieller Käufer bzw. Auftraggeber wird zunächst nach Angeboten suchen, bevor er sich bei Interesse mit dem Anbieter in Verbindung setzt. Dabei kann es sein, dass er gezielt nach bestimmten Angeboten sucht oder wahllos Angebote anschaut (im letzten Fall möchte er sich dann eher einen Überblick verschaffen, welche Art von Angeboten in dem System veröffentlicht wurden). Interessiert ihn ein Angebot, so wird er sich dieses näher ansehen wollen, das heißt, er wird sich über Beschreibung, Preis etc. informieren und die zugehörigen Bilder begutachten. Er wird hierbei abwägen, ob sich das Angebot mit seinen Bedürfnissen bzw. seiner Zahlungsbereitschaft deckt und wird sich ggf. auch Informationen über den Anbieter verschaffen. Hierbei interessiert ihn möglicherweise, wer der Anbieter ist (Privatperson, kleine Firma, große Firma, großes Unternehmen etc.), wie lange das Mitglied bereits in dem System registriert ist, wie es sich selbst beschreibt etc.

Ist das Mitglied dann immer noch an dem Angebot interessiert, so wird es in einem nächsten Schritt Kontakt mit dem Anbieter aufnehmen, genau wie im Fall 1 bereits beschrieben wurde. Die beiden Mitglieder werden schließlich einen Vertrag abschließen und den Tausch ausführen. Spätestens dann wird der Kunde die vereinbarte Summe an den Anbieter zahlen. Für ihn endet der Geschäftsprozess, sobald das Geld überwiesen und die Leistung erbracht ist.

1.3.4 Fall 3: Mitglieder registrieren

Im Fall der Anhalt Dessau AG werden Mitglieder wie folgt registriert: Ein Interessent füllt ein Anmeldeformular aus und wird dann als Kontakt im System registriert. Hierbei wird ihm ein Kundenbetreuer zugewiesen, der diesen betreut. Möchte sich der Kontakt vollständig im System registrieren, um Angebote erstellen und wahrnehmen zu können, so muss der Kontakt ein weiteres Formular ausfüllen und wird

im Anschluss vom Kundenbetreuer als Kunde registriert. Erst dann ist die Person bzw. das Unternehmen verbindliches Mitglied im System und hat volle Rechte und Pflichten.

Die Aufgaben des Kundenbetreuers sind u.a. die Vermittlung von Angeboten sowie die Beratung, Unterstützung und Verwaltung der ihm zugeordneten Mitglieder.

1.4 Anforderungen

1.4.1 Grundanforderungen

In diesem Abschnitt sollen nun die Anforderungen erläutert werden, die für ein Software-System vorliegen, das in regionalen Entwicklungs-Netzwerken eingesetzt wird. Diese sollen möglichst allgemein gehalten werden, da sich die einzelnen Unternehmen, wie bereits deutlich geworden sein sollte, teilweise erheblich von den Funktionalitäten und Rahmenbedingungen her unterscheiden. Als Grundlage für die Anforderungen dienen die beiden vorangegangen Abschnitte, in denen die Regionalentwicklungs-Netzwerk erläutert und die Geschäftsvorfälle aufgezeigt wurden.

Wie bereits am Beispiel der Anhalt Dessau AG gezeigt wurde, ist Kooperation ein ganz wesentliches Merkmal von Regionalentwicklungsunternehmen. Kooperationsplattformen bilden gewissermaßen die Basis für alle Funktionalitäten, die angeboten werden. Das bedeutet, dass eine Software-Lösung zunächst einmal die Möglichkeit bieten muss, dass Mitglieder des Unternehmens auf dieser Plattform miteinander in Beziehung treten und verschiedene Aktionen ausführen können. Im Grunde genommen ist damit bereits eine erste grundlegende Anforderung geklärt: Es muss sich um ein webgestütztes Softwaresystem handeln. Die Mitglieder müssen schließlich die Möglichkeit haben, in umfangreichem Maße miteinander interagieren zu können. Alles andere als eine Web-Plattform wäre daher nicht problemangemessen.

Das einzusetzende Softwaresystem muss natürlich übersichtlich und gut bedienbar sein. Dies ist in der Softwaretechnik unterdessen eine grundlegende Anforderung, die eigentlich gar keiner Erwähnung bedarf; nichtsdestoweniger werden auch heute noch Softwaresysteme entwickelt, die diesem Standard nicht in voller Weise gerecht werden. Da Personen aus allen Branchen und mit unterschiedlichsten Computerkenntnissen die Plattform nutzen sollen, sind also gute Bedienbarkeit, Übersichtlichkeit und ein leicht verständliches Benutzerhandbuch (bzw. eine integrierte Hilfe-Anwendung) von hoher Bedeutung. Das Layout der Plattform ist hingegen von geringerer Bedeutung. Aufwendige grafische Gestaltungen sind sicherlich nicht das Primärziel eines solchen Softwaresystems und sollten eher entfallen.

Zu einer guten Bedienbarkeit zählt natürlich auch die Tatsache, dass die Softwa-

re in der entsprechenden Landessprache verwendet werden kann. Eine Lösung, die beispielsweise nur in englischer Sprache angezeigt wird, würde sonst von den meisten Unternehmen in einem nicht-englischsprachigen Land nicht eingesetzt werden können.

Weitere wichtiger Aspekt bei Softwaresystemen, die mit Ersatzwährungen arbeiten und den Tausch von Waren und Dienstleistungen ermöglichen, sind Sicherheit und Zuverlässigkeit. Da auf Kooperationsplattformen auch größere Beträge überwiesen werden können (im Grunde existiert nach oben hin oft keine Beschränkung), ist es von hoher Bedeutung, dass alle Transaktionen absolut sicher durchgeführt und in allen Einzelheiten protokolliert werden. Immerhin dienen die Protokolle auch als Basis für eine eventuelle rechtliche Auseinandersetzung zwischen Mitgliedern. Damit verbunden ist auch die Verwendung eines leistungsfähigen Datenbanksystems (z.B. MySQL, DB2) mit sicherer Transaktionsverwaltung, guter Skalierbarkeit und möglichst geringen Zugriffszeiten, da in dem System mehrere zehntausend Mitglieder integriert sein können.

1.4.2 Benutzerfunktionen

Angebote erstellen

Jedes Mitglied muss in der Lage sein, beliebig viele Angebote und Gesuche erstellen zu können, wobei Angebote und Gesuche grundlegend voneinander zu trennen sind. Im Anschluss daran muss das Bearbeiten und Löschen des Angebots zu jedem Zeitpunkt gewährleistet werden. Es wurde bereits gezeigt, welche Eigenschaften ein Angebot besitzen muss:

- Titel
- Name des Anbieters
- Beschreibung
- Mehrere Bilder
- Preis
- Gültigkeitszeitraum

Das Verwenden von Videos, die im vorangegangenen Abschnitt kurz angesprochen wurde, kann relativ einfach umgesetzt werden, indem Benutzer die Möglichkeit besitzen, Videodateien hochzuladen, die vom Webbrowser ausgeführt werden können.

Nach dem Erstellen muss das Angebot für alle anderen Mitglieder sichtbar werden, es muss also auf der Kooperationsplattform veröffentlicht werden.

Angebote suchen

Jedes Mitglied muss nach Angeboten suchen können. Wie schon erwähnt, muss eine direkte Suche nach Angeboten ermöglicht werden sowie eine allgemeine Suche, um sich einen Überblick über vorhandene Angebote verschaffen zu können. Im ersten

Fall muss das System eine leistungsfähige Suchfunktion bieten, so dass nicht nur nach Titel, sondern auch nach Beschreibung, Preisspanne, Zeitraum und Anbieter gesucht werden kann. Hier wäre auch eine Volltextsuche vorteilhaft. Im zweiten Fall ist es eher sinnvoll, alle Angebote in Kategorien (Branchen) zusammenzufassen. Da es sehr viele Wirtschaftszweige und Branchen gibt, wären Unterkategorien ebenfalls sinnvoll. Angebote sollten also schon zu Beginn einer Branche zugeordnet sein, deren Zuordnung der Ersteller vornimmt. Auch eine Funktion, die alle vorhandenen Angebote auflistet, wäre durchaus empfehlenswert.

Nachdem die Suche durchgeführt wurde, müssen die gefundenen Resultate aufgelistet werden. Hierbei sollte der Anwender auch die Sortierreihenfolge ändern können (z.B. Sortierung nach Preis, Zeitspanne, Kategorie etc.).

In der Resultatliste sollte bereits der Name des Angebots, die Beschreibung (oder zumindest ein Teil derselbigen) und ein Bild des Angebots sichtbar sein, so dass der Interessent schnell entscheiden kann, welche Angebote relevant erscheinen und welche nicht. Natürlich muss es dann auch möglich sein, das gesamte Angebot mit allen Angaben einsehen zu könne.

Damit sich ein Interessent auch nähere Informationen zu dem Anbieter verschaffen kann, muss jedes Mitglied über ein Profil verfügen, so wie es bei den meisten Web-Plattformen unterdessen üblich ist. Die genauen Angaben einer Person bzw. eines Unternehmens (z.B. Name, Adresse, Branche, Beschreibung, Registrierdatum etc.) können natürlich zwischen den einzelnen Entwicklungsgesellschaften verschieden sein; ein Softwaresystem sollte daher das Hinzufügen bzw. Entfernen von Profilattributen unterstützen.

Um einen Anbieter bzw. einen Interessenten besser einschätzen zu können, kann ein Bewertungssystem nützlich sein. Solche Systeme bieten allerdings auch die Gefahr, dass subjektive oder unangemessene Bewertungen vergeben werden und können dann eher schädlich als nützlich sein. Da vermutlich wenig Unternehmen ein Bewertungssystem verwenden werden, ist diese Funktion daher als optional einzustufen.

Kommunikation

Bevor ein Vertrag abgeschlossen wird, werden die Mitglieder möglicherweise miteinander in Kontakt treten, um sich über Details zu beraten, den Preis auszuhandeln, Sonderwünsche zu besprechen etc. Hierfür müssen Kommunikationsmittel bereitstehen, wobei ein einfacher, im System integrierter Nachrichtendienst als völlig angemessen erscheint. Die Nachrichten sollten dann wie E-Mails mit einem Betreff, einem Adressaten, einem Nachrichtenbereich sowie einigen Zusatzangaben (z.B. Absendezeit) ausgestattet sein.

Das Verwenden eines internen Chats wäre eine mögliche Ergänzung des Nachrichtendienstes, ist aber eher als optional einzustufen. Zwar können somit Angebote einfacher und schneller besprochen werden, hierzu müssen jedoch Anbieter und Interessent gleichzeitig online sein, was möglicherweise nicht oft der Fall sein wird.

Vertrag abschließen

Der Vertrag ist im Prinzip wie das Angebot aufgebaut. Allerdings sind hierbei alle Angaben verbindlich, das heißt, Preis, Konditionen, Lieferzeitpunkt etc. sind eindeutig geregelt und dürfen nach Vertragsunterzeichnung nicht mehr geändert werden können.

Beide Vertragspartner müssen dem Vertrag zustimmen, bevor dieser gültig wird. Solange der Vertrag noch nicht gültig ist, muss er sowohl vom Anbieter als auch vom Kunden zurückgenommen werden können. Ist er jedoch abgeschlossen, so darf er nicht mehr geändert werden und muss revisionssicher gespeichert werden. Natürlich müssen beide Mitglieder zu jedem Zeitpunkt die Möglichkeit besitzen, den Vertrag einsehen zu können.

Bezahlung

Die Bezahlung sollte per Überweisung innerhalb des Systems stattfinden und vom Empfänger der Ware oder Leistung durchgeführt werden. Wie bereits in den vorhergehenden Abschnitten dargestellt, müssen Split-Beträge ermöglicht werden. Da Unternehmen wie die Anhalt Dessau AG auch über mehrere Währungen verfügen, ist es erforderlich, die Bezahlung in mehreren Währungen zu ermöglichen; die Währungen sollten dann zur Vergleichbarkeit über einen Wechselkurs miteinander in Beziehung stehen. Da sich das Währungssystem in einer regionalen Entwicklungsgesellschaft ändern kann, wäre es sinnvoll, dass das Hinzufügen, Löschen und Ändern von Währungen ermöglicht wird.

Weitere Funktionen

Eine Kooperationsplattform zur Unterstützung von Regionalentwicklungsgesellschaften sollte auch noch einige weitere Funktionalitäten bieten, die nur indirekt aus den Geschäftsvorfällen hervorgehen. Dazu zählen u.a.

- Registrieren
- Einloggen und Ausloggen
- Ändern der persönlichen Daten im Profil
- Überblick über den Kontostand und durchgeführte Transaktionen

Beim Registrieren kann es mehrere Formen geben. In manchen Fällen wird sich eine Person registrieren und sofort vollständiges Mitglied sein, in anderen Fällen muss ein Admin den Account zunächst freischalten. Am Beispiel der Anhalt Dessau AG zeigt sich ein weiterer Fall: Hier werden neue Mitglieder zunächst als Kontakt betrachtet. Es wird die Adresse des Kontakts gespeichert und ihm ein Kundenbetreuer zugeordnet. Der Kundenbetreuer wandelt den Kontakt zu einem späteren Zeitpunkt ggf. in ein vollständiges Mitglied (Kunde) um. Das Softwaresystem muss somit auch diesen Fall berücksichtigen.

Zudem sollte eine grundlegende Unterteilung der Mitglieder in Adminstratoren und einfache Mitglieder erfolgen, so wie es auf den meisten Web-Plattformen üblich ist. Das Beispiel der Anhalt Dessau AG zeigt jedoch auch, dass Kundenbetreuer berück-

sichtigt werden müssen, die funktionell zwischen Administratoren und Mitgliedern stehen. Insofern sollte das Softwaresystem also diese 3 Arten unterstützen.

1.4.3 Administrative Funktionen

Da in den vorangegangenen Abschnitten bereits gezeigt wurde, dass sich die einzelnen Regionalentwicklungs-Netzwerke in Funktionsweise und Aufbau teils erheblich voneinander unterscheiden, sollte der Administrator die Möglichkeit besitzen, grundlegende Programmkonfigurationen vornehmen zu können. Dazu zählt z.B. das Anlegen neuer Währungen und Branchen, das Ändern der Eigenschaften von User-Profilen und Angeboten, das Ändern des Firmenlogos sowie einiger Formateinstellungen (Datum, Uhrzeit, Schriftart etc.).

1.4.4 Zusammenfassung

Im Nachfolgenden sollen die grundlegenden funktionalen Anforderungen noch einmal schematisch dargestellt werden.

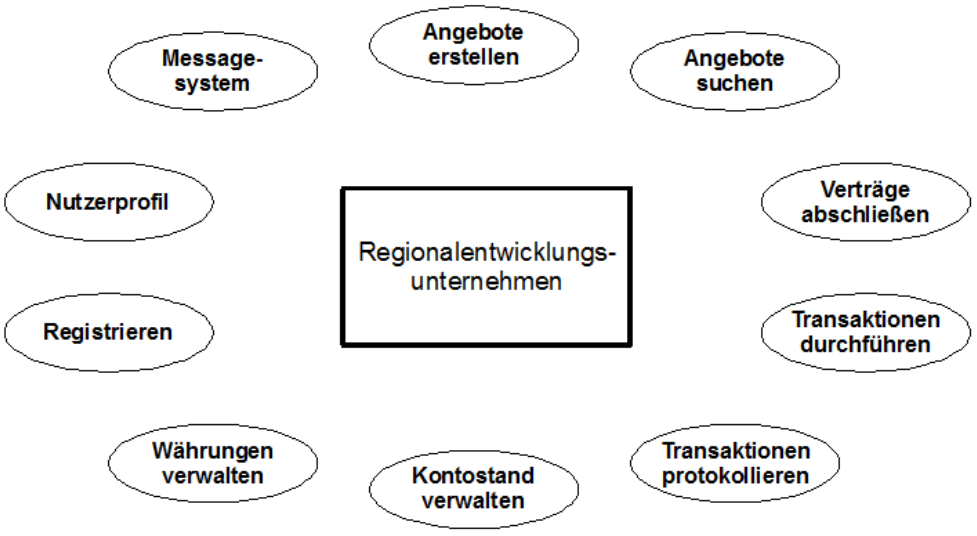

Abbildung 1: Grundlegende funktionale Anforderungen an ein Softwaresystem zur Unterstützung regionaler Entwicklungsnetzwerke.

2 Architektur von Web-Plattformen

2.1 Einleitung

Bevor die beiden Software-Plattformen Cyclos und OLAT vorgestellt und näher untersucht werden sollen, soll ein allgemeiner Überblick vermittelt werden, wie Webanwendungen grundlegend aufgebaut sind. Natürlich gibt es hierbei z.T. deutliche Abweichungen zwischen den zahlreichen Webapplikationen, die mittlerweile im World Wide Web eingesetzt werden, es gibt allerdings auch grundlegende Strukturen und Richtlinien, die von den meisten Anwendungen eingehalten werden. Für Webanwendungen scheinen mehrere Programmiersprachen geeignet zu sein, z.B. HTML, CSS, JavaScript etc. In den letzten Jahren haben sich jedoch PHP und Java am meisten durchgesetzt, weil sie einen mächtigen Funktionsumfang bieten und vor allem das Erstellen dynamischer Websites unterstützen. Da OLAT und Cyclos beidermaßen in Java geschrieben sind, soll der Fokus in diesem Kapitel auf Webanwendungen liegen, die in Java geschrieben sind.

2.2 Grundlagen

Webanwendungen sind für den direkten Einsatz im World Wide Web konzipiert und werden auf einem Web-Server ausgeführt. Nutzer (Clients) bedienen Webanwendungen von ihrem Browser aus, so dass sich eine Client-Seite und eine Server-Seite ergibt. Der Benutzer kann innerhalb der Webanwendungen verschiedene Funktionen ausführen. Man unterscheidet hierbei zwei Arten von Funktionen: Das Verfolgen eines Links (GET-Request) und das Klicken eines Buttons (POST-Request). Links dienen dazu, um den Benutzer auf eine andere Seite zu navigieren bzw. die Seite zu aktualisieren. Buttons dienen hingegen dazu, um ein bestimmtes Formular abzuschicken und damit eine bestimmte Aktion durchzuführen. Unter einem Formular versteht man dabei nicht nur herkömmliche Eingabeformulare mit Textfeldern und Textareas, sondern auch alle anderen Eingabe- und Auswahlkomponenten wie Listen, Comboboxen, Radiobuttons, Checkboxen, Tabellen etc.

Wird ein Link verfolgt oder ein Button geklickt, so wird ein GET- bzw. POST-Request vom Client an den Server gesendet sowie alle zugehörigen Parameter (z.B. die Formulardaten, falls ein Formular abgeschickt wurde). Die Übertragung der Da-

ten geschieht mittels dem HTTP-Protokoll. Die Applikation wird dann über die Aktion informiert (z.B. über ein Servlet) und erhält die übermittelten Daten. Die Daten werden entsprechend verarbeitet und es wird ggf. eine neue Seite generiert bzw. zu einer anderen Seite weitergeleitet. Dazu wird ein HTTP-Response zurück an den Client gesendet, mit dem vom Browser die Ausgabe erstellt wird, so wie sie die Webapplikation dynamisch erzeugt hat. Man erkennt dabei, dass Webanwendungen also ebenfalls nach dem EVA-Prinzip (Eingabe - Verarbeitung - Ausgabe) funktionieren.

Webanwendungen enthalten dynamische und statische Dateien. Dynamische Dateien werden erst zur Laufzeit generiert und sind meist HTML-Dateien, die während einer Benutzer-Session individuell erzeugt bzw. angepasst werden. Statische Dateien sind Dateien, die niemals geändert werden und bereits in fertiger Form auf dem Server liegen. Dazu zählen z.B. CSS-Stylesheets, Bilder, Icons, Hilfe-Dateien, Konfigurationsdateien und natürlich alle Libraries und Programmklassen.

Da Webapplikationen ein hohes Maß an Sicherheit benötigen und stets mehrere Benutzer gleichzeitig auf die Anwendung zugreifen können, verwenden sie meist relationale Datenbanksysteme zur Speicherung und Verwendung persistenter Daten. In den Tabellen werden in erster Linie Benutzerdaten gespeichert, aber auch Programmkonfigurationen und Einstellungen, und teilweise sogar Programmkomponenten wie Formulare mit den entsprechenden Attributen (Feldern), Wertebereichen, Parametern etc.

Viele Webanwendungen nutzen auch einige Frameworks, die den Implementierungsaufwand deutlich reduzieren können. Einige bedeutende Frameworks sind Spring, Struts und Hibernate, die später noch vorgestellt werden sollen, da sie auch von OLAT bzw. Cyclos verwendet werden.

2.3 3-Schichten-Architektur

Große Softwareprodukte, und insbesondere auch Webanwendungen, werden oft in mehrere Schichten aufgeteilt, so wie es auch in anderen Bereichen der Informatik üblich ist, z.B. bei Datenbanksystemen oder beim IP-Protokoll. Dabei nutzt eine Schicht n immer nur die Dienste und Funktionen der darunter liegenden Schichten.

Besonders häufig wird eine 3-Schichten-Architektur (engl. 3-Tier-Architecture) verwendet, wonach die Software in 3 Schichten eingeteilt wird: Präsentation und Benutzerinteraktion (Client Tier), Anwendungs- und Geschäftslogik (Middle Tier) und Datenschicht (EIS Tier) [10, S. 1, S. 4, S. 254].

Die Präsentationsschicht stellt die Schnittstelle zum Benutzer dar. Sie dient zur Ein-

und Ausgabe von Daten und damit zur direkten Interaktion mit dem Anwender. Die darunter liegende Anwendungs- und Geschäftslogik dient zur Berechnung sowie zur Manipulierung von Daten. Sie kann als Mittler zwischen der oberen und unteren Schicht angesehen werden, da sie einerseits auf die Datenbank zugreift, um Daten ändern zu können, und andererseits die entsprechenden Daten der Präsentationsschicht zur Verfügung stellt, damit diese sie ausgeben kann. In der Datenverwaltung werden schließlich sämtliche Daten gehalten, die für den Programmbetrieb benötigt werden. Dies geschieht üblicherweise mittels einer Datenbank und einer Anbindung über Hibernate.

Die 3-Schichten-Architektur ist unterdessen ein allgemeines Konzept, das sich in der Softwaretechnik etabliert hat. Die meisten Softwaresysteme lassen sich jedoch meist noch in feinere Schichten untergliedern, wobei die genaue Zuordnung einer solchen feineren Schicht zu einer der drei o.g. Schichten oft nicht ganz eindeutig ist. Die Grenzen können also fließend sein. Unabhängig von den einzelnen Schichten wird von vielen Webanwendungen konsequent das MVC-Konzept umgesetzt (Model, View, Controller), das später noch genauer beschrieben werden soll.

2.4 Verzeichnisstruktur einer Webapplikation

Eine Webapplikation besitzt eine hierarchische Struktur, die durch ein herkömmliches Dateisystem oder eine Archivdatei realisiert werden kann (im Folgenden soll von einem Dateisystem ausgegangen werden) [10, S. 40 ff.]. Oft gibt es ein Verzeichnis *webapps*, welches die eigentliche Web-Applikation repräsentiert. Statt webapps wird manchmal auch der Name des Programms verwendet (also z.B. cyclos, olat).

Innerhalb dieses zentralen webapps-Verzeichnisses gibt es i.A. einen WEB-INF-Ordner und ggf. noch einen META-INF-Ordner. Hierbei handelt es sich um besondere Verzeichnisse, da die darin enthaltenen Dateien nicht direkt an den Client gesendet werden und nicht über HTTP erreicht werden können.

Das WEB-INF-Verzeichnis enthält typischerweise die gesamten Programmklassen sowie die Bibliotheken und Konfigurationsdaten der verwendeten Frameworks. Das META-INF-Verzeichnis wird verwendet, falls die Webapplikation als Archivdatei verteilt werden soll und enthält die Manifestdatei des Archivs und ggf. eine Signaturdatei [10, S. 42 f.]. Das WEB-INF-Verzeichnis untergliedert sich meist in einen Ordner *classes*, worin das Java-Klassenverzeichnis enthalten ist, einen Ordner *lib*, *libs* oder *libraries*, in dem sämtliche verwendete Bibliotheken liegen (meist sind diese zu einem jar zusammengefasst) und ggf. noch weitere Ordner, in denen Programmdaten liegen (z.B. *tags* für Tag Files, *themes* für Theme-Dateien etc.). Manchmal wird die Klassenhierarchie auch zu einem jar zusammengefasst und liegt im Library-Verzeichnis; der classes-Ordner ist dann womöglich leer.

In dem WEB-INF-Verzeichnis liegt zudem die Datei web.xml, die grundlegende Einstellungen zu der Webanwendung enthält.

Da die Verzeichnisse META-INF und WEB-INF nicht über HTTP erreichbar sind, liegen statische Dateien wie Bilder, Icons, HTML-Dokumente, CSS-Dateien i.A. nicht in diesen Verzeichnissen, sondern direkt im webapps-Verzeichnis. Dort gibt es meist einen Ordner für Bilder (z.B. images), für CSS-Stylesheets (z.B. css), für die integrierte Hilfe (z.B. help) etc.

Außerhalb des webapps-Verzeichnis können Konfigurations- und Datenbankdaten liegen, die Programmdokumentation, Lizenzen, ReadMe- und Info-Dateien etc.

2.5 Das MVC-Konzept

Die meisten komplexen Softwareprodukte, insbesondere aber Webanwendungen, werden heute nach dem MVC-Prinzip (Model, View, Controller) erstellt. Dieses hat sich in den letzten Jahren zu einer Art Standard entwickelt und wird auch bei den beiden Kooperationsplattformen eingehalten, die Gegenstand dieser Arbeit sind.

Bei dem MVC-Konzept wird die Anwendung in drei logische Teile gegliedert [10, S. 15 f.]. Das View dient zur Erstellung der Ein- und Ausgabe und ist damit die direkte Schnittstelle zum Anwender. In herkömmlichen Anwendungen wurde das View oft mit einem bestimmten Framework realisiert, z.B. AWT oder Swing bei Java. Bei Webanwendungen verwendet man heute oft JSP-Pages (Java Server Pages) oder Java Servlets, die dynamischen HTML-Code generieren.

Während das View den Zustand bestimmter Objekte darstellt und somit zur Ein- und Ausgabe verwendet wird, dient das Model zur Manipulation von Objekten. Zum Model gehören somit alle Funktionen, die Objekte in irgendeiner Form ändern. In den meisten Fällen handelt es sich dabei um Berechnungsmethoden, Zugriffsmethoden (get) und Änderungsmethoden (set, delete, add etc.). Da komplexe Softwaresysteme üblicherweise Datenbanken zur Verwaltung der Daten verwenden, gehören somit auch alle DB-Zugriffe zum Model.

Der Controller kann als Mittler zwischen Model und View angesehen werden. Ruft der Benutzer eine Funktion auf, so wird der Browser-Request meist von einem Servlet bzw. Dispatcher des Programms abgefangen. Diese rufen einen entsprechenden Controller auf, der für die Bearbeitung des GET- oder POST-Requests zuständig ist. Für jede Funktion sollte dabei genau ein Controller zur Verfügung stehen (manchmal stehen aber auch mehrere Controller zu einer Funktion zur Verfügung). Die Aufgabe des Controllers ist es nun, das korrekte Model aufzurufen, welches zur Durchführung

der gewünschten Aktion verantwortlich ist. Das Model führt dann die eigentliche Aktion aus, nicht der Controller. Im Anschluss wird das View benachrichtigt, welches die Ausgabe in HTML generiert und an den Browser des Clients zurücksendet.

2.6 Einige wichtige Frameworks

Zuletzt sollen noch einige wichtige Frameworks vorgestellt werden, die oft in Webanwendungen zum Einsatz kommen. Dies sind vor allem:

- Hibernate
- Spring
- Struts

Hibernate und Spring kommen dabei in OLAT und Cyclos gleichermaßen zum Einsatz; Struts wird hingegen nur von Cyclos verwendet.

Hibernate ist ein Framework zur Anbindung einer relationalen Datenbank an ein Softwaresystem [11]. Wie bereits erwähnt, ist es heute üblich, sämtliche Daten mit einem relationalen Datenbanksystem zu verwalten, z.B. MySQL, das kostenlos verwendet werden kann. Um in einem objektorientierten Programm mit der Datenbank komfortabel arbeiten zu können, müssen die in den Tabellen der Datenbank enthaltenen Daten in Objekte transformiert werden, was oft einen sehr hohen Arbeitsaufwand bedeutet. Das Hibernate-Framework ermöglicht hierbei, dass das Portieren der Daten aus der Datenbank bzw. zurück in die Datenbank mit möglichst wenig Aufwand geschieht. Hierbei wird zu jeder Relation eine (Java-) Klasse angelegt, welche dieselben Attribute wie die zugehörige DB-Relation besitzt. Anschließend werden ihr sämtliche Getter und Setter hinzugefügt und es wird eine hbm.xml-Datei erstellt, in der die Bezeichner und notwendigen Parameter (Datentyp etc.) der Relation und der Klasse eingetragen werden. Im Anschluss daran können die Daten von Hibernate selbstständig in das Programm geladen und in die Datenbank zurückgeschrieben werden.

Spring ist eine Java/J2EE-Plattform, die eine Vielzahl von Funktionen und Tools zur Erstellung von Java-Anwendungen bietet [12]. So gibt es beispielsweise Java Beans zur einfachen Konfiguration der Applikation, einen JDBC Abstraction Layer, der das Exception-Handling von SQL-Exceptions erleichtern soll, die Integration anderer Frameworks wie Hibernate und JDBC, sowie ein integriertes MVC-Framework zur Erstellung von Webanwendungen.

Struts ist ein OpenSource-Framework, das die Realisierung des MVC-Konzepts in einer Web-Applikation unterstützt. Es wurde von der Apache Software Foundation entwickelt und kann kostenlos verwendet werden. Struts unterstützt u.a. mittels Tag-Libraries und dem integrierten Tiles-Framework das Erstellen des Views sowie

das komfortable Verarbeiten von Formularen mittels ActionForm Beans [13] [10, S. 164 f.]. Das Framework wird bei der technischen Analyse von Cyclos noch näher erläutert.

3 Analyse von Cyclos

3.1 Einleitung

Cyclos ist ein Open Source Softwareprojekt, das vor allem von Regionalgeld-Gesellschaften eingesetzt werden kann und eine Standardlösung für die im ersten Kapitel vorgestellten regionalen Entwicklungsgesellschaften bilden soll. Das Software-System bietet dabei alle grundlegenden Funktionen, die für Tauschgeschäfte bzw. Transaktionen auf Basis von Ersatzwährungen notwendig sind. Damit dürfte Cyclos für viele Unternehmen von potentiellem Interesse sein, da es kostenlos beschafft und verwendet werden kann und zudem auch die Möglichkeit zur individuellen Erweiterung bzw. Anpassung bietet.

Das Cyclos-Projekt wurde im Februar 2003 gestartet und wird momentan von 7 Entwicklern weiterentwickelt. Es wird heute bereits von vielen Unternehmen eingesetzt, u.a. auch von RegioSTAR und Regio Geld Urstromtaler.[15]

In diesem Kapitel sollen der Funktionsumfang und die Softwarearchitektur von Cyclos näher untersucht werden. Dabei wird zunächst die Installation von Cyclos erläutert, der Funktionsumfang und anschließend der Aufbau des Programms auf Software-Ebene sowie die Möglichkeiten der Erweiterung und Anpassung von Cyclos.

Für diese Arbeit wurde die Version Cyclos 3.0.9 verwendet, die bereits relativ stabil läuft. Es gibt auch bereits neuere Versionen, bspw. Version 3.5, bei der es jedoch bei der Installation und der Ausführung noch einige Komplikationen gegeben hatte. Daher soll in dieser Arbeit nur die Version 3.0.9 betrachtet werden.

[15] http://project.cyclos.org/index.php?option=com_content&task=view&id=233&Itemid=236

3.2 Installation

Cyclos kann von der Cyclos-Website (http://project.cyclos.org/) kostenlos heruntergeladen werden. Dabei werden mehrere Cyclos-Versionen angeboten. Zu jeder Programmversionen existieren bis zu drei verschiedene Download-Versionen:

- Stand Alone Installation (Quick Installation)
- Basis Installation Package
- Upgrade Package

Die Stand Alone Quick Installation wird für die meisten Windowsversionen (ab Windows 98) und alle Linuxversionen angeboten.[16] Hierbei handelt es sich um ein ca. 47 MB großes Zip-Archiv (im Fall der Windows-Version), das lediglich eine Installationsdatei zum Installieren des Programms enthält. Somit kann Cyclos komfortabel installiert und dann sofort ausgeführt werden; insbesondere müssen keine weiteren Programme (Java, MySQL, Tomcat etc.) installiert werden. Nach dem Entpacken der Zip-Datei kann Cyclos sofort durch Starten der Datei cyclos.bat ausgeführt werden. Das Programm wird standardmäßig im Webbrowser unter http://localhost:8080/ ausgeführt.

Das Basis Installation Package enthält das gesamte Cyclos Package mit dem WEBINF-Ordner, allen Libraries und Konfigurationsdateien sowie dem gesamten Source Code. Diese Downloadversion ist zu wählen, falls die Programmarchitektur näher analysiert werden oder erweitert werden soll. Voraussetzung ist dabei eine installierte Java-Version (mindestens Java 1.5, jedoch nicht Java 1.6.0_00 bis 1.6.0_03), MySQL 4.1 oder höher und der Apache Tomcat 5.0 oder höher.[17]

Das Upgrade Package ist nur für den Upgrade einer älteren Cyclos-Version vorgesehen und enthält lediglich die Dateien, die im Vergleich zur Vorgängerversion neu hinzugekommen sind bzw. geändert wurden. Damit soll ein recht einfaches Upgraden einer bereits im Einsatz befindlichen Cyclos-Version ermöglicht werden.

Für diese Arbeit wurde das Cyclos 3.0.9 Basis Installation Package verwendet. Um das Programm ausführen zu können, muss zunächst eine Datenbank *cyclos* in MySQL angelegt werden. Anschließend wird der web-Ordner aus dem heruntergeladenen Verzeichnis in den webapps-Ordner des Tomcat-Verzeichnisses kopiert und (optional) in cyclos3 umbenannt. Im Anschluss daran muss die Datei cyclos.properties, die im Verzeichnis cyclos3/WEBINF/classes/ liegt, konfiguriert werden. Hierbei muss der Pfad zum Programmordner und zur Datenbank eingetragen bzw. angepasst werden. Startet man dann den Tomcat, so wird das Programm gestartet. Bei erstmaliger Ausführung wird dabei die Datenbank initialisiert, was ein wenig Zeit in Anspruch nehmen kann.

[16] http://project.cyclos.org/index.php?option=com_content&task=view&id=220&Itemid=219
[17] http://project.cyclos.org/wiki/index.php?title=Installation_%26_maintenance

3.3 Funktionsumfang

3.3.1 Allgemeine Funktionen

In diesem Abschnitt sollen allgemeine Funktionen und die im System verwendeten Mitgliederarten vorgestellt werden, bevor dann in den nächsten beiden Abschnitten die genauen Funktionen der Mitglieder und Administratoren näher erläutert werden sollen.

Layout und Sprache

Cyclos ist ein webbasiertes Anwendungsprogramm, das über einen Webbrowser ausgeführt wird. Es wird zwischen drei unterschiedlichen Mitgliederarten unterschieden, auf die später noch genauer eingegangen werden soll. Nach dem Einloggen wird der Account des jeweiligen Members angezeigt. Dabei gibt es eine Kopfzeile mit dem Cyclos-Logo und grundlegenden Informationen (Zeitpunkt des Logins etc.) und darunter folgend einen Menüleisten-Bereich auf der linken Seite sowie einem Ausgabe- bzw. Funktionsbereich auf der rechten Seite. Die Menüleiste besteht aus mehreren Menüpunkten (in Abhängigkeit der Art des eingeloggten Benutzers), die bei Auswahl meist eine Reihe von Untermenüs aufdecken. Wird ein Untermenü angeklickt, so ändert sich i.A. der Ausgabebereich und man hat die Möglichkeit, eine bestimmte Aktion auszuführen.

Cyclos kann in mehreren Sprachen verwendet werden. Wie in Java üblich, sind die einzelnen Programmtexte in verschiedenen Dateien gespeichert und werden über einen bestimmten Key vom Programm aus angesprochen. Die Software wird bereits in den Sprachen US-Englisch, Spanisch, Portugiesisch, Deutsch, Italienisch, Japanisch und Niederländisch angeboten. Da jede Sprache in nur einer Datei verwaltet wird (die sog. ApplicationRessource-Datei), die relativ logisch aufgebaut ist, ist das Hinzufügen neuer Sprachen sehr einfach möglich und erfordert im Grunde nur das Kopieren und anschließende Übersetzen einer bereits vorhandenen Application-Ressource-Datei.

Mitgliederarten

Im Cyclos wird grundsätzlich zwischen einem Member und einem Admin unterschieden. Dies erkennt man bereits daran, dass Member und Admins einen grundsätzlich verschiedenartigen Menüaufbau besitzen, der auch in unterschiedlichen Dateien gespeichert ist.

Member sind allgemeine Mitglieder und werden den größten Anteil in einem Unternehmen bilden, das Cyclos verwendet. *Admins* dienen zum Verwalten der Member sowie zum Konfigurieren von Cyclos.

Member können von einem Admin das Broker-Recht gewährt bekommen und werden dann als *Broker* bezeichnet. Broker besitzen neben den Member-Funktionen noch

die Möglichkeit, neue Mitglieder im System zu registrieren und zu verwalten. Sie können also als Member mit administrativen Fähigkeiten angesehen werden, werden aber intern noch immer als Member betrachtet und nicht als Admin. Im Gegensatz zu Administratoren haben Broker keinen Einfluss auf die Programm-Konfiguration bzw. die zahlreichen Konfigurationsparameter; außerdem müssen von einem Broker angelegte Member stets noch einmal von einem Administrator freigeschaltet werden.

Member, Broker und Admins sind noch einmal in spezifischere Gruppen, so genannte *Permission Groups*, untergliedert. Eine solche Gruppe legt genaue Rechte und Einstellungen für eine Benutzergruppe fest. Bei den Membern gibt es somit z.B. Pending Members, Full Members, Disabled Members, Removed Members etc. Ein Full Member ist dann ein Mitglied mit vollen Rechten, ein Disabled Member ist ein (temporär) gesperrtes Mitglied etc. Der Administrator kann die Einstellungen zu den jeweiligen Permission Groups vornehmen und neue Permission Groups anlegen bzw. bereits vorhandene löschen.

Hilfe und Support

Die Software bietet bereits ein integriertes Hilfesystem, das bei der Version 3.0.9 aber noch etwas spärlich erscheint. Es findet nur eine knappe Erläuterung der Grundfunktionen statt; als ein vollständiges, ausführliches Benutzerhandbuch kann man das Hilfesystem daher nicht ansehen. Allerdings ist es möglich, in jedem Programmmenü, zu dem ein Hilfe-Eintrag existiert, direkt über einen Hilfe-Button zu dem entsprechenden Hilfe-Eintrag zu gelangen. Damit wird eine höhere Benutzerfreundlichkeit und Benutzerunterstützung bewirkt. Außerdem scheint das Hilfesystem in der Version 3.5 bereits deutlich umfangreicher zu sein.

Währungen

Cyclos bietet alle grundlegenden Funktionen, die für Tauschgeschäfte mit Regionalgeld benötigt werden. Im Cyclos gibt es die Standardwährung *units*, die für alle Bezahlvorgänge verwendet werden kann. Ein Währungsobjekt besitzt dabei einen Namen, eine Beschreibung, eine Abkürzung und ein Symbol. Hierbei ist zu erkennen, dass kein Attribut *wechselkurs* existiert. Werden mehrere Währungen verwendet, ist also ein Vergleich der Währungen nicht möglich.

Der Administrator kann im Cyclos grundsätzlich weitere Währungen anlegen. Auffällig ist dabei, dass die neu angelegten Währungen jedoch nicht im Programm verwendet werden können. Das heißt, Mitglieder können für Bezahlungen nach wie vor nur die Währung *units* verwenden, auch wenn neue Währungen angelegt wurden. Es ist jedoch anzunehmen, dass es sich hierbei um einen Programmfehler oder eine bewusst noch nicht vollendete Funktionalität handelt, so dass dieser Missstand mit den nächsten Cyclos-Versionen wahrscheinlich behoben sein wird.

Branchen

Neben der Unterstützung von Währungen sind auch Branchen für teilnehmende Unternehmen eine wichtige Notwendigkeit. Daher bietet Cyclos die Möglichkeit,

beliebig viele Branchen zu verwenden. Der Administrator kann Branchen anlegen, bearbeiten und löschen. Eine Branche besitzt dabei einen Namen und eine Beschreibung. Zudem kann der Administrator Branchen aktivieren und deaktivieren und Unterkategorien anlegen. Eine weitere Untergliederung (z.B. Anlegen von Unter-Unter-Kategorien) wird allerdings nicht unterstützt. Zudem zeigt sich schnell, dass bei entsprechend vielen Branchen bzw. Unterkategorien das Auflisten der Branchen unübersichtlich wird, da Unterkategorien weder eingerückt, noch ausgeblendet werden können.

3.3.2 Funktionen des Admins

Der Administrator hat zwei wesentliche Funktionen: Verwaltung von Mitgliedern und Verwaltung bzw. Kontrolle des Programmbetriebs. Er hat dabei sowohl einen hohen Einfluss auf die Mitglieder, als auch auf die Programmkonfiguration.

Der Administrator hat einen vollen Überblick über alle im System eingeloggten und registrierten Mitglieder. Er kann deren Kontostand einsehen, erhaltene Bewertungen, erstellte Angebote und ausgeführte Transaktionen. Er kann auch Mitglieder ausloggen und ihre Rechtegruppe ändern (das Verschieben eines Mitglieds in die Permission Group *Removed Member* gleicht somit dem Löschen eines Members) und er kann das Passwort eines Mitglieds ändern, so dass er theoretisch auch Zugang zu dem Account hat. Ganz offensichtlich verfügt der Administrator also über einen hohen Einfluss auf Mitglieder und Broker.

Zudem haben Admins die Möglichkeit, Cyclos durch Ändern von Programmparametern anzupassen. Hierbei zeigt sich, dass Cyclos enorm konfigurierbar ist. So können die Profil-Felder angepasst werden (z.B. Ändern der Sichtbarkeit oder des Wertebereichs der Eingabe, Hinzufügen oder Löschen von Eingabefeldern), die Sprache geändert, das Datums- und Zahlenformat festgelegt, Währungen und Branchen hinzugefügt oder gelöscht, und zahlreiche weitere Programmeinstellungen vorgenommen werden.

3.3.3 Funktionen des Members

Member besitzen ein Profil, das anderen Mitgliedern (Member, Broker, Admins) zugänglich ist. Dort können dann verschiedene Funktionen ausgeführt werden, z.B. das Senden einer Nachricht, das Überweisen von Geldbeträgen etc. Zudem entspricht das Profil einer Art Visitenkarte, auf dem einige Angaben zu dem Member ersichtlich sind, z.B. Name, Unternehmen etc.

Eine der wichtigsten Funktionalitäten ist natürlich das Veröffentlichen von Angeboten (Advertisements, kurz Ads), wobei Cyclos grundsätzlich zwischen Angebot und Gesuch unterscheidet (im Nachfolgenden sollen unter dem Begriff *Angebot* auch

Gesuche mit eingeschlossen sein). Ein Angebot besitzt dabei einen Titel, eine Beschreibung, eine Branche, einen Preis und ggf. einen Zeitraum, in dem es gültig ist. Titel und Beschreibung sind dabei obligatorische Angaben, die anderen Felder sind optional. Zusätzlich können auch mehrere Bilder hochgeladen werden. Die maximale Anzahl an Bildern wird durch den Administrator begrenzt.

Die Angebote eines Mitglieds werden übersichtlich in seinem Account angezeigt und können nachträglich bearbeitet und auch wieder gelöscht werden. Zudem kann ein Mitglied nach Angeboten und Gesuchen anderer Mitglieder suchen. Hierfür steht eine allgemeine Suchfunktion bereit. Durch Eingabe entsprechender Stichworte werden alle relevanten Angebote gesucht und die Ergebnisse in einer Liste ausgegeben. Die Auflistung mehrerer Ergebnisse erfolgt alphabetisch, wobei das Ändern der Sortierreihenfolge nicht möglich ist. Cyclos bietet auch noch ein erweitertes Suchmenü an, indem eine spezifischere Suche nach Angeboten ermöglicht wird. Hierbei kann ggf. eine Kategorie ausgewählt und eine Preisspanne festgelegt werden. Zudem kann nach Angeboten gesucht werden, die innerhalb der letzten Tage eingestellt wurden (die genaue Spanne ist variabel festlegbar) sowie gezielt nach Angeboten, die Bilder enthalten.

Nützlich in Verbindung mit Angeboten sind sog. Interests (Inserate). Ein Inserat besteht dabei aus mindestens einem Suchwort, das auf ein bestimmtes Angebot passen soll. Sobald dann von einem Member ein neues Angebot veröffentlicht wird, das auf das entsprechende Inserat passt, wird der Benutzer über eine Nachricht informiert. Dabei kann das Inserat noch feiner angepasst werden. So können auch noch ein bestimmtes Mitglied, eine Preisspanne und eine Kategorie festgelegt werden. Außerdem kann wieder nach Angebot und Gesuch unterschieden werden.

Wie bereits im ersten Kapitel erwähnt, ist die Kommunikation unter Mitgliedern ein essentieller Bestandteil von regionalen Entwicklungsnetzwerken. Cyclos bietet hierfür u.a. folgende Funktionalitäten:

- Suchen nach Mitgliedern
- Kontaktliste
- Nachrichtendienst
- Bewertung

Das Suchen nach Mitgliedern erfolgt über ein einfaches Suchmenü. Es kann entweder nach dem Member-Namen (Account-Name) oder dem richtigen Namen (Vor- und Nachname) gesucht werden. Die Ergebnisse werden dann, genau wie die Angebote, in einer Liste ausgegeben. Klickt man einen Eintrag in der Liste an, so gelangt man zu dem Profil des Members, wo man, wie schon erwähnt, einige Aktionen in Bezug zu dem Member ausführen kann.

Die Kontaktliste ermöglicht das Speichern beliebig vieler Member. Somit kann direkt aus der Kontaktliste auf das Profil der Member zugegriffen werden. Im Gegensatz zu vielen anderen Netzwerken, z.B. meinvz.net, ist die Liste allerdings unilateral. Das heißt, in der Kontaktliste eingetragene Member wissen nicht, dass sie dort eingetragen sind.

Der im Cyclos integrierte Nachrichtendienst funktioniert im Prinzip wie auf solchen Web-Plattformen allgemein üblich ist: Es gibt eine Inbox für empfangene Nachrichten, eine Outbox für versendete Nachrichten und eine Box für gelöschte Nachrichten. Nachrichten können an alle Member bzw. Admins und Broker versendet werden und haben neben einem Nachrichtentext einen Adressaten (Membername) und einen Betreff. Außerdem muss ausgewählt werden, ob die Nachricht an ein Member bzw. Broker gesendet werden soll oder an einen Administrator. In letzterem Fall muss noch einmal eine bestimmte Kategorie ausgewählt werden (z.B. Support, Beschwerde, etc.). Die Kategorien können vom Administrator angepasst und verwaltet werden.

Eine wichtige Funktionalität ist natürlich auch das Bezahlen von Leistungen. Hierzu gibt es insgesamt vier verschiedene Kombinationen, die sich aus zwei verschiedenen Kriterien ergeben:

- Beteiligte Mitglieder (Member oder System)
- Art der Bezahlung (Rechnung oder Überweisung)

Im Allgemeinen werden Member-zu-Member-Bezahlungen durchgeführt, das heißt, eine Bezahlung findet zwischen zwei Membern statt (Member-Payment). Es besteht aber auch die Möglichkeit, dass eine Bezahlung zwischen einem Member und einem Admin durchgeführt wird (z.B. zur Erhalt/Rückzahlung eines Kredits, zur Bezahlung einer Gebühr etc.). Letztere Bezahlungen heißen System-Payments (also Bezahlungen zwischen Member und System). Bezahlungen finden stets zwischen genau zwei Personen statt, natürlich kann aber jede Rechnung, an der mehr als 2 Personen beteiligt sind, durch mehrere Bezahl-Vorgänge realisiert werden.

Soll nun eine Bezahlung durchgeführt werden, so gibt es genau zwei Möglichkeiten: Entweder über eine Rechnung oder durch Überweisung.

Wie in der realen Welt wird eine Rechnung vom Auftragnehmer an den Auftraggeber (Kunden) gesendet. Dieser kann sie annehmen oder ablehnen. Der Auftragnehmer erhält dann eine Nachricht, wenn der Kunde eine der beiden Aktionen durchgeführt hat. Zudem kann er die Rechnung auch zurückziehen, sofern sie nicht bereits angenommen und bezahlt wurde.

Bei einer Überweisung geht die Bezahlung direkt vom Kunden aus. Er überweist einen bestimmten Betrag an den Auftraggeber. Dieser bekommt das Geld direkt auf sein Konto überwiesen.

Bezahlungen werden stets in units durchgeführt, das Verwenden mehrerer Währungen (sog. Split-Beträge) ist, wie bereits erwähnt, momentan noch nicht möglich. Rechnungen und Überweisungen können zudem noch einen Beschreibungstext enthalten, damit der Sender bzw. Empfänger erfährt, worauf sich die Bezahlung bezieht.

Mitglieder können andere Mitglieder auch bewerten („References“). Dabei kann eine Bewertungsstufe ausgewählt werden (very good, good, neutral, bad, very bad) und ggf. eine Beschreibung. Jedes Mitglied kann ein anderes Mitglied nur einmal bewerten, jedoch nachträglich ggf. die Bewertung bzw. Beschreibung ändern.

Neben diesen grundlegenden Funktionen gibt es noch eine Reihe von zusätzlichen Funktionen. So besteht die Möglichkeit, sich den Kontostand, die erhaltenen und vergebenen Bewertungen und zahlreiche weitere Account-Informationen anzuzeigen. Es kann zu jeder grundlegenden Funktion ein Hilfeeintrag angesehen werden und es besteht die Möglichkeit, einige Nachrichten-Einstellungen vorzunehmen.

3.4 Technischer Aufbau

3.4.1 Einführung

Cyclos ist Open Source Software, die von zwei Programmier-Teams mit Sitz in Brasilien und Uruguay weiterentwickelt und von STRO Uruguay and Instrodi [18] betreut wird. Die Software ist weitgehend in Java geschrieben, wobei die Ausgabe mittels Java Server Pages und JavaScript realisiert wird. Für die Datenbank wird MySQL verwendet, das über das Hibernate-Framework an Cyclos angebunden ist.

In diesem Abschnitt soll der technische Aufbau von Cyclos näher untersucht und dargelegt werden. Hierzu zählen vor allem das Funktionsprinzip von Cyclos auf Programmebene, die Strukturierung des Programms und Quellcodes, die Aufgabe und Funktionsweise der verwendeten Frameworks (insbesondere Struts und Tiles) sowie die Erweiterungsmöglichkeiten.

3.4.2 Verzeichnisstruktur von Cyclos

Das Cyclos-Verzeichnis besteht aus einem web-Ordner, in dem alle Programm- und Konfigurationsdateien sowie alle benötigten Bibliotheken von Cyclos enthalten sind. Wie bereits erwähnt, wird dieses Verzeichnis bei der Installation ggf. umbenannt (z.B. in *cyclos*) und in das webapps-Verzeichnis des Tomcats kopiert und von dort ausgeführt.

Der cyclos-Ordner ist zunächst in vier Verzeichnisse untergliedert: *mobiles*, *wap*, *pages* und *web-inf*. Die Verzeichnisse mobiles und wap enthalten nur einige wenige Konfigurationsdateien, die für den mobilen Dienst von Cyclos benötigt werden (z.B. Zugriff auf Cyclos per Handy etc.). Diese Dienste werden aber in der Version 3.0.9 noch nicht unterstützt und werden erst in späteren Versionen Bestandteil von Cyclos sein; daher soll hierauf nicht weiter eingegangen werden. Der pages-Ordner kann als das View von Cyclos aufgefasst werden. Hierin enthalten sind alle JSP-Pages (Java Server Pages) sowie alle Java-Script-Dateien. Diese werden von Cyclos

[18] http://project.cyclos.org/index.php?option=com_frontpage&Itemid=1

zur Generierung der Ausgabe benötigt. In dem web-inf-Verzeichnis liegen schließlich der eigentliche Programmordner sowie sämtliche Konfigurationsdateien, Frameworks und Libraries. Insbesondere sind hierin also auch die Controller und das Model enthalten. Abbildung 2 zeigt noch einmal den Aufbau des Cyclos-Verzeichnis sowie die Unterverzeichnisse des web-inf-Ordners.

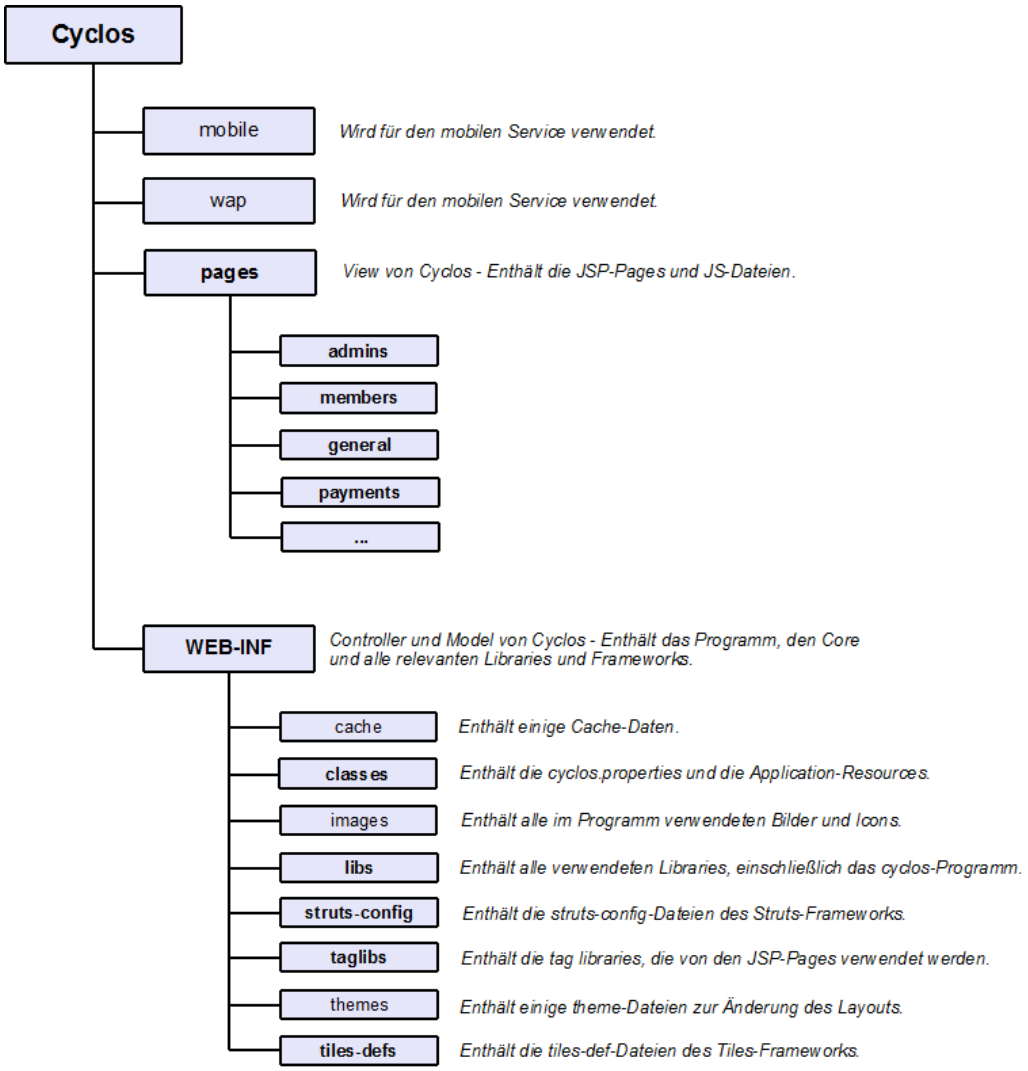

Abbildung 2: Verzeichnisstruktur von Cyclos.

Im cache-Verzeichnis befinden sich einige Cache-Daten, die vom Cyclos während des Programms angelegt und ggf. verwendet werden, um z.B. Zugriffe auf die Datenbank zu reduzieren.

Im classes-Verzeichnis liegt die Datei *cyclos.properties*, die zur grundlegenden Konfiguration von Cyclos verwendet wird. Hier werden u.a. der Pfad zum Programm eingetragen, der Pfad zur Datenbank sowie grundlegende Programm- und Datenbankeinstellungen (z.B. Verwendung von SSL etc.). Außerdem befinden sich hierin die ApplicationRessource-Dateien, in denen die Texte und Bezeichnungen für alle

Label, Felder, Messages etc. eingetragen sind, die innerhalb des Programms verwendet werden (für jede Sprache gibt es hierbei eine separate Datei). Anders als der Name des Verzeichnisses vermuten lässt, liegen im classes-Verzeichnis nicht die Programmklassen. Diese werden im Cyclos zu einem jar-Archiv zusammengefasst und in das lib-Verzeichnis eingefügt. Nichtsdestoweniger ist es auch möglich, Cyclos auszuführen, indem man das ungepackte Klassenverzeichnis von Cyclos in den classes-Ordner kopiert.

Im images-Verzeichnis liegen die im Programm verwendeten Images (Icons, Bilder etc.).

Das lib-Verzeichnis ist das zentrale Verzeichnis, das alle Bibliotheken enthält und den zu einem jar zusammengefassten Programmordner. Hierin enthalten sind auch die Bibliotheken der verschiedenen Frameworks, von SQL, Java, dem Tomcat etc.

In den Verzeichnissen struts-config und tiles-def liegen die einzelnen Konfigurationsdateien des Struts-Frameworks und des Tiles-Frameworks. Diese nehmen eine besondere Rolle im Cyclos ein und sollen später noch genauer erläutert werden.

Im tag-libs-Verzeichnis liegen einige tag-Bibliotheken. Diese werden von den JSP-Pages verwendet und enthalten grundlegende Funktionen zur Generierung der Ausgabe, gehören also mit zur View-Schicht.

Im themes-Verzeichnis liegen mehrere theme-Dateien, mit denen das grundlegende Layout von Cyclos angepasst werden kann.

Weiterhin ist in dem web-inf-Verzeichnis die Datei web.xml enthalten, die Teil des Struts-Frameworks ist und ebenfalls später noch näher erläutert werden soll.

3.4.3 Quellcode

Der Programmordner von Cyclos ist /nl/strohalm/cylos/ und untergliedert sich dann in weitere Verzeichnisse. Er ist etwa 4,3 MB groß und besteht aus etwa 1022 Java-Klassen mit insgesamt rund 70.000 Zeilen Quellcode. Im Vergleich zu anderen Programmen, die teilweise mehrere Millionen LOC Umfang haben können, ist Cyclos 3.0.9 somit relativ überschaubar.

Der Quellcode erfüllt weitgehend die grundlegenden Code-Konventionen, wie sie bspw. in [14] beschrieben werden. Zwar wird die vorgeschriebene Zeilenlänge von 80 Zeichen oftmals überschritten, die Bezeichner, Methoden und die Strukturierung des Quellcodes sowie des Programmordners entsprechen aber etwa den allgemeinen Regeln. Es fällt jedoch schnell auf, dass das Programm kaum dokumentiert ist. Nur die wichtigsten Klassen und Methoden sind beschrieben, dann jedoch meist unvollständig (meist keine Verwendung der allgemeinen JavaDoc-Tags). Auch Inline-Kommentare sind selten, so dass es sehr aufwendig ist, sich in den Programmcode

einzuarbeiten bzw. die Funktionsweise und Bedeutung einzelner Klassen zu erschließen.

Dieser Nachteil wird durch ein Wiki, das u.a. vom Programm-Team erstellt wurde, etwas ausgeglichen. In dem darin enthaltenen Programm-Wiki werden grundlegende Komponenten des Programms, die grundlegende Funktionsweise sowie die Erweiterungsmöglichkeiten erläutert. Allerdings werden hier nur einzelne Aspekte näher beschrieben, so dass es weiterhin recht schwierig ist, sich einen Gesamtüberblick über das System zu verschaffen. Das umfangreiche Wiki kann daher die unzureichende Dokumentation nicht vollständig ausgleichen; die Einarbeitung in das Programm ist entsprechend aufwendig.

3.4.4 Workflow

In diesem Abschnitt soll die Programmarchitektur von Cyclos erläutert werden. Diese lässt sich besonders gut am Beispiel des Workflows veranschaulichen, weil hierbei alle Schichten des Programms und alle wesentlichen Klassen durchlaufen werden. Auch die verwendeten Frameworks (Spring und Struts) lassen sich damit gut erklären.

Browser-Requests

Der Workflow beginnt, wenn im Browser eine bestimmte Aktion gestartet wird und endet, wenn diese vollständig ausgeführt ist. Eine Aktion kann z.B. das Anzeigen aller Währungen oder das Bearbeiten einer Währung sein, wobei diese Aktionen in diesem Abschnitt oft als Beispiel dienen sollen.

Sobald eine solche Aktion gestartet wurde, muss ein entsprechender Controller die Aktion verarbeiten. Hierfür müssen im Grunde zwei Dinge bekannt sein: Die genaue Aktion, die ausgeführt werden soll, und der zu dieser Aktion zugehörige Controller zur weiteren Bearbeitung. Dies wurde im Cyclos mit dem Struts-Framework realisiert.

Action-Weiterleitung mittels Struts

Struts ist ein Framework auf Basis der MVC-Struktur, das im Cyclos eine zentrale Rolle einnimmt. Es handelt sich hierbei um eine Schnittstelle, die sich zwischen Browser und Controller befindet. Struts nimmt dabei einen Actionkey vom Browser entgegen, ermittelt den zu dieser Action gehörenden Controller und gibt die Kontrolle dann an den entsprechender Controller ab. Struts leitet also Browser-Requests an die entsprechende Controller-Klasse weiter.

Zentrale Bestandteile des Struts-Frameworks sind:

- Die Datei web.xml
- Die struts-config-Dateien
- Die Datei struts-config.xml

Die Datei web.xml liegt im Verzeichnis WEB-INF und ist die zentrale Konfigurationsdatei der Web-Plattform. Sie enthält grundlegende Konfigurationen, die Links zu den verwendeten Servlets und Spring-Dateien sowie die Willkommensseite (bzw. Startseite) von Cyclos (index.jsp). Außerdem enthält die Datei die Links zu den struts-config-Dateien, die für das Weiterleiten des Browser-Requests an den entsprechenden Controller verantwortlich sind.

Die struts-config-Dateien haben folgendes Format: struts-config_*name*.xml, wobei *name* der Name einer bestimmten Funktionsgruppe ist. Jede dieser Dateien enthält eine oder mehrere Action-Tags. Jedes Action-Tag dient zum Mappen einer bestimmte Aktion. Die Datei struts-config_currencies.xml dient beispielsweise zur Weiterleitung aller währungsspezifischen Aktionen. Dabei gibt es im Cyclos genau 3 Stück: *listCurrencies*, *editCurrencies* und *removeCurrencies*. Folglich gibt es in der Datei 3 Action-Mappings. Das nachfolgende Beispiel soll zeigen, wie das ActionMapping der Funktion listCurrencies aussieht:

```
<action
    path="/admin/listCurrencies"
    type="nl.strohalm.cyclos.controls.accounts.currencies.ListCurrenciesAction"
    input="admin/_listCurrencies">
    <set-property property="secure" value="true" />
</action>
```

Dabei sind vor allem die Attribute *path* und *type* von besonderer Bedeutung. Hierbei gibt *path* den Schlüssel der Aktion an, so wie ihn Struts vom Browser erhält und *type* ist der exakte Pfad zu der Java-Klasse (Controller), welche die entsprechende Aktion verarbeiten soll. Man erkennt somit, dass diese Action-Tags die Browser-Requests und Controller verbinden.

Ähnlich sehen auch die Action-Mappings von *editCurrency* und *removeCurrency* aus, die ebenfalls in struts-config_currencies.xml eingetragen sind. Hierbei handelt es sich allerdings um Aktionen, die ein Formular verwenden. Als Formular zählt dabei jegliche benutzerseitige Eingabe (z.B. Eingabe in Textfeldern, aber auch Checkboxen, Listen, Comboboxen etc.). Actions mit Formulardaten enthalten ein weiteres Attribut *name*, welches den Namen des Formular-Controllers angibt (dazu später mehr).

Wenn nun im Cyclos eine Aktion ausgewählt wird, so wird dies über das zum Struts-Framework zugehörige ActionServlet registriert. Es wird nach dem entsprechenden Action-Tag gesucht, der zur Weiterleitung des Requests dient. Die Action-Tags werden bereits zu Programmstart geladen, so dass die Suchzeit sehr gering ist. Sobald der entsprechende Tag gefunden wurde, wird die unter *type* eingetragene Java-Klasse aufgerufen. Hierbei handelt es sich stets um einen Controller, der für die weitere Verarbeitung der Aktion zuständig ist.

Controller

Im Cyclos gibt es zu jeder elementaren Funktion eine Action-Klasse, die im Package *controls* enthalten ist. Falls die Aktion Formulardaten verwendet, so gibt es zudem noch eine Form-Klasse. Die Funktion *listCurrencies* verwendet kein Formular und besitzt nur die Controller-Klasse *ListCurrenciesAction*. Die Funktion *editCurrencies* verwendet hingegen Formulardaten und besitzt neben der Klasse *EditCurrencyAction* noch die Klasse *EditCurrencyForm*. Für den Workflow sind jedoch nur die Action-Klassen interessant, welche die eigentlichen Controller darstellen und die Form-Klassen nur als Hilfsmittel für die Verarbeitung der Formulardaten benötigen.

Jede Action-Klasse ist von einer der folgenden drei Klassen abgeleitet:

- BaseAction
- BaseFormAction
- BaseQueryAction

Dabei gilt folgendes Prinzip: Suchfunktionen werden von BaseQueryAction abgeleitet, sofern sie Parameter enthalten; andernfalls werden sie direkt von BaseAction abgeleitet. Edit-Funktionen, also Funktionen, bei denen etwas bearbeitet wird (z.B. *EditCurrency*), sind von BaseFormAction abgeleitet. Bei Remove-Funktionen (Funktionen, bei denen etwas gelöscht wird, z.B. *RemoveCurrencies*) ist der Controller direkt von BaseAction abgeleitet.[19]

Die Klassen BaseQueryAction und BaseFormAction sind von der zentralen Klasse BaseAction abgeleitet. Jeder Controller ist somit direkt oder indirekt von dieser Klasse abgeleitet. In BaseAction liegt die zentrale Methode *execute()*, die folglich von jeder Controller-Klasse geerbt wird. Dabei fällt auf, dass *execute()* als final definiert ist; sie darf also von unterliegenden Klassen nicht überschrieben werden.

Sobald Struts den entsprechenden Controller ermittelt hat (so wie er im Action-Mapping eingetragen ist), ruft das Framework die Methode *execute()* des entsprechenden Controllers auf. Soll z.B. die Funktion *listCurrencies* ausgeführt werden, so durchsucht Struts die Action-Mappings und findet im Mapping *listCurrencies* die Klasse ...controls.accounts.currencies.ListCurrenciesAction. Struts ruft nun die execute()-Methode dieser Klasse auf.

Der execute()-Methode werden einige Attribute überreicht: Das ActionMapping, das ActionForm, ein HTTPServletRequest und ein HTTPServletResponse. Das ActionMapping enthält die Action-Parameter, wie sie im Struts angegeben sind (type, path etc.); das ActionForm enthält die Formulareingaben des Users (sofern ein Formular existiert). Innerhalb der Methode wird nun der User validiert. Hierbei wird z.B. geprüft, ob der User noch eingeloggt ist und ob es sich um einen gültigen Request handelt. Im Anschluss daran wird ein ActionContext-Objekt erstellt. Dieses hat 6 Attribute: Das ActionMapping, das ActionForm, den http-Request, den http-Response, den eingeloggten User und eine ServiceFetch-Instanz. Letztere dient dazu, um auf Relationships von Entitäten zugreifen zu können.[20]

[19] http://project.cyclos.org/wiki/index.php?title=Programming_guide#Struts_actions
[20] http://project.cyclos.org/wiki/index.php?title=Programming_guide#Lazy_loading_relationships

Das ActionContext-Objekt kann also gewissermaßen als eine Hülle angesehen werden, die um alle notwendigen Daten gespannt wird, die für die weitere Verarbeitung notwendig sind. Es fasst alle Informationen zusammen, die die Controller und Service-Klassen benötigen. Dabei ist *ActionContext* von der abstrakten Klasse *AbstractActionContext* abgeleitet.

Nachdem das ActionContext-Objekt erstellt wurde, wird die Methode *executeAction()* aufgerufen. Dabei muss executeAction() von BaseAction überschrieben werden und ist somit ein Erweiterungspunkt. Die Methode bekommt das ActionContext-Objekt übergeben und führt die gewünschte Aktion schließlich aus. Jede Controller-Klasse besitzt eine solche Methode, die bereits in BaseAction als abstrakte Methode angelegt ist. Ist eine Action-Klasse bspw. von BaseAction abgeleitet, so überschreibt sie executeAction() direkt. Das nachfolgende Beispiel zeigt die Methode executeAction() der Klasse ListCurrenciesAction.

```
protected ActionForward executeAction(ActionContext context) throws Exception {
    HttpServletRequest request = context.getRequest();
    List<Currency> currencies = currencyService.listAll();
    request.setAttribute("currencies", currencies);
    request.setAttribute("editable", getPermissionService().checkPermission("
        systemCurrencies", "manage"));
    return context.getInputForward();
}
```

Etwas anders verhält es sich mit Action-Klassen, die nicht von BaseAction abgeleitet sind. Diese sind dann folglich von BaseFormAction oder BaseQueryAction abgeleitet, und jede dieser beiden Klassen überschreibt die Methode executeAction(). Dort ist executeAction() als final definiert, so dass es den unterliegenden Action-Klassen nicht möglich ist, diese zu ändern. Es werden im Anschluss verschiedene Methoden zur Validierung der Formulardaten aufgerufen, die eine Verfeinerung des Erweiterungspunktes executeAction() sind. Das nachfolgende Beispiel zeigt die Methode executeAction() der Klasse BaseFormAction.

```
    protected final ActionForward executeAction(ActionContext context) throws Exception {
        if (isFormPreparation(context)) {
            return handleDisplay(context);
        } else if (isFormValidation(context)) {
            return handleValidation(context);
        } else if (isFormSubmission(context)) {
            return handleSubmit(context);
        } else {
            return context.sendError("errors.invalid_request");
        }
    }
```

Die von der Methode aufgerufenen Methoden liefern einen bestimmten ActionForward zurück, der Auskunft darüber gibt, ob die Aktion erfolgreich durchgeführt wurde oder nicht. Die Methode *handleSubmit()* liefert bspw. bei korrekter Ausführung *ActionForward.getSuccessForward()* zurück, kennzeichnet also eine erfolgreich durchgeführte Aktion. Die einzelnen Methoden (z.B. *handleValidation()*, *handleSubmit()* etc.) müssen jeweils von der entsprechenden Controller-Klasse überschrieben werden; sie sind bereits als abstrakte Methoden in BaseFormAction und BaseQueryAction vordefiniert.

Die Methode *getSuccessForward()* liegt ebenfalls in AbstractActionContext und liefert zurück, ob die Aktion erfolgreich war. Sie hat folgenden Aufbau:

```
public ActionForward getSuccessForward() {
    return actionMapping.findForward("success");
}
```

Die Methode *findForward()* liegt in der Klasse ActionMapping, die Teil des Struts-Frameworks ist. Sie dient also dazu, den entsprechenden Forward zu finden, so dass das Tiles-Framework im Anschluss die Ausgabe generieren kann.

Jede executeAction()-Methode liefert ein ActionForward-Objekt zurück. Anschließend gibt die Controller-Klasse die Kontrolle an Struts ab (genauer: an das Tiles-Framework, das aber Teil von Struts ist) und die Ausgabe wird generiert.

Aktionsausführung durch die Service-Klassen

Wie bereits in dem Quellcode-Beispiel der Methode executeAction() der Klasse ListCurrenciesAction zu erkennen ist, findet die eigentliche Aktionsverarbeitung nicht im Controller statt. Wie in Java üblich gibt es hierfür eine Model-Klasse. Der Controller dient nur dazu, die entsprechende Model-Klasse aufzurufen und die Ergebnisse entgegenzunehmen. Hierbei wird ein HttpServletRequest-Objekt erstellt und nach Ausführung der Model-Klasse werden die Ergebnisse auf diesem Objekt gespeichert, wobei dafür die Methode *setAttribute()* zuständig ist. Auf dem Objekt sind somit die Ergebnisse gespeichert, die später beim Generieren der Ausgabe benötigt werden.

Für die eigentliche Ausführung der Aktionen sind die Model-Klassen zuständig, die im Cyclos als Services bzw. Service-Klassen bezeichnet werden. Zu jeder Funktionsgruppe gibt es ein Service-Interface und eine implementierende Service-Klasse. Jede Klasse enthält dabei alle Funktionen, die in Verbindung mit der Funktionsgruppe möglich sind (hierbei existiert für jede elementare Funktion i.A. genau eine Methode).

Für die Währungen gibt es somit ein Interface CurrencyService und eine implementierende Klasse CurrencyServiceImpl. Dabei enthalten die Klassen alle währungsspezifischen Funktionen sowie einige Hilfsmethoden. Wie in dem nachfolgenden Quellcode-Beispiel ersichtlich, wird vom Controller eine entsprechende Methode in der CurrencyService-Klasse aufgerufen, z.b. *listAll()* zum Anzeigen aller Währungen.

```
public List<Currency> listAll() {
    if (cachedCurrencies == null) {
        cachedCurrencies = currencyDao.listAll();
    }
    return cachedCurrencies;
}
```

In der Methode *listAll()* werden also die bereits im Cache vorhandenen Währungen an den Controller zurückgeliefert. Liegen diese nicht im Cache, so müssen sie aus der Datenbank angefordert werden. Es ist im Cyclos allerdings nicht möglich, dass eine Service-Klasse direkt auf die Datenbank zugreift. Dies geschieht stets mittels eines Hilfsobjekts, den sog. DAO-Objekten.

Entitäten, DAO's und Datenbank

Cyclos nutzt zur Verwaltung der Daten MySQL, wobei die Daten in 67 Tabellen abgelegt sind. Um die Daten direkt im Programm nutzen zu können, verwendet Cyclos das Hibernate-Framework und eine Reihe von Entity-Klassen (Entitäten). Diese Entity-Klassen repräsentieren genau eine Datenbank-Tabelle und werden über das Hibernate mit dieser verbunden. Solche Klassen besitzen lediglich einige Attribute (i.A. genau die Attribute der Tabelle) und entsprechende Getter und Setter bzw. Konstruktoren. Mit jeder Entity sind dann verschiedene Aktionen verbunden. Mit der Entity *Währung* sind z.B. die Funktionen *listCurrencies*, *editCurrency* und *removeCurrency* verbunden.

Wie bei den Services gibt es zu jeder Entity ein DAO-Interface und eine implementierende DAO-Klasse, die von *BaseDaoImpl* abgeleitet ist. Diese DAO-Klassen (Data Access Object) werden von den Service-Klassen verwendet, um direkt auf die Datenbank zugreifen zu können. Die DAO-Klassen können also als Schnittstelle zwischen Services (Model) und Datenbank aufgefasst werden, gehören aber logisch zur Model-Schicht und nicht zur Datenbank.

Das nachfolgende Beispiel zeigt die Methode *listAll()* der Klasse *CurrencyDAOImpl*, die die Währungen aus der Datenbank ausliest.

```
public List<Currency> listAll() {
    return list("from Currency c order by c.name", null);
}
```

BaseDaoImpl bietet dabei zahlreiche nützliche Funktionen für Datenbankanfragen, z.B. *list()*, *delete()*, *update()*, *insert()* etc. Somit kann in einer bestimmten DAOImpl-Klasse komfortabel auf die Datenbank zugegriffen werden (insbesondere werden Quellcode und lange Anfragen durch die vordefinierten Funktionen in BaseDaoImpl eingespart).

Das Ergebnis wird nun zu der Controller-Klasse zurückgereicht. Man erkennt an diesem Workflow, dass nachfolgendes Schichtensystem (Abbildung 3) existiert, das im Cyclos bei der Ausführung eines Requests durchlaufen wird. Bei der Skizze muss jedoch berücksichtigt werden, dass sich das View nur auf das Tiles-Framework bezieht; das Action-Mapping, das von Struts durchgeführt wird, gehört nicht zum View, sondern eher zum Controller (genauer formuliert ist es die Schnittstelle zwischen Browser und Controller).

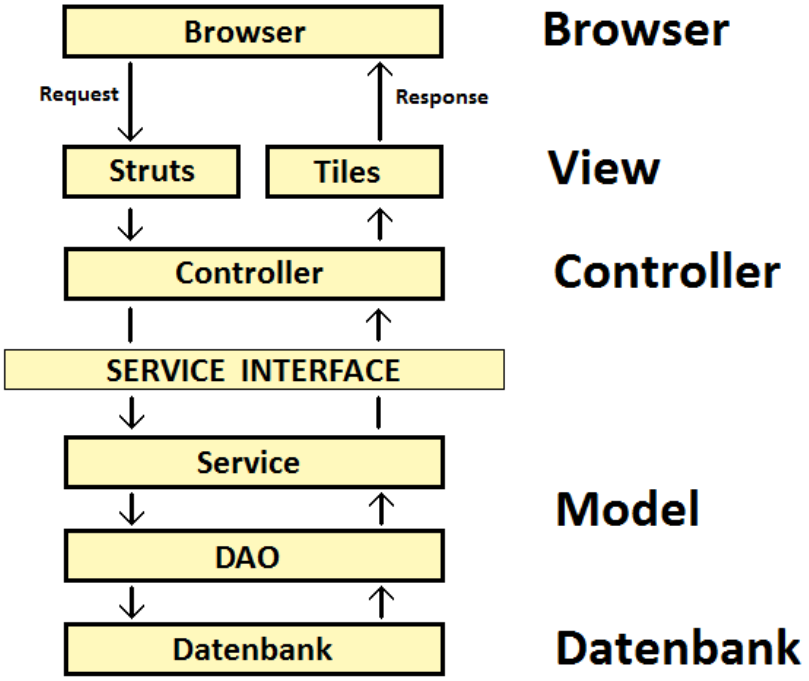

Abbildung 3: Schichtenarchitektur von Cyclos und Workflow.

Generieren der Ausgabe

Die Ausgabe geschieht mittels JSP-Pages, die zusammen mit JS-Dateien das View von Cyclos bilden. Dabei ist es im Cyclos eine allgemeine Konvention, Java Script nicht in den JSP-Pages zu verwenden; somit existiert meist zu jeder JSP-Page eine JS-Datei mit gleichem Namen. Von hoher Bedeutung sind dabei u.a. die Dateien *menuAdmin.jsp* und *menuMember.jsp*, die sich im Verzeichnis pages/general/layout/ befinden. In den beiden Dateien sind die jeweiligen Menüpunkte des Admin-Accounts bzw. des Member-Accounts eingetragen.

Das Tiles-Framework, das Teil des Struts-Frameworks ist, dient nun dazu, von einem Controller zu der entsprechenden JSP-Page weiterzuleiten. Es funktioniert ähnlich wie das Struts und besteht aus einer Reihe von tiles-def-Dateien, in denen mehrere *definitions* angelegt werden (analog zum Struts, in dem in mehreren struts-config-Dateien actions angelegt werden). Jede tiles-def-Datei umfasst dabei definitions einer bestimmten Funktionsgruppe, für die das Generieren einer Ausgabe notwendig ist. Im Falle der Währungen betrifft dies die Funktionen editCurrency und listCurrencies. Da bei removeCurrency keine neue Ausgabe generiert wird, existiert zu dieser Funktion keine definition.

Das folgende Beispiel zeigt die definition zu der Funktion listCurrencies an, so wie sie in der Datei tiles-def_currencies.xml angelegt ist.

```
<definition name="admin/_listCurrencies" extends=".adminLayout">
    <put name="body" value="/pages/accounts/currencies/listCurrencies.jsp"/>
</definition>
```

Das Attribut *name* gibt dabei den Namen der Aktion an, die soeben ausgeführt wurde. Das Framework kennt diesen Namen, weil der Controller ihn an Tiles zurückliefert. Das Attribut *extends* gibt ein Layout an, das bei der Generierung der Ausgabe verwendet (bzw. geerbt) werden soll. Diese erbbaren Layout sind auch wieder definitions, die in der Datei tiles-def.xml angelegt sind. Interessanter ist jedoch der put-Tag. Hierbei gibt das Attribut *value* die JSP-Page an, die ausgeführt werden soll, um die entsprechende Ausgabe zu generieren. Zu jeder elementaren Funktion gibt es genau eine solche JSP-Page. Das Attribut *name* gibt den Bereich auf der Website an, der geändert werden soll und ist stets *body* (also der Teil rechts neben dem Menü).

Man erkennt hierbei, dass Tiles vom Prinzip her nicht anders als Struts funktioniert. Genau wie Struts bildet Tiles die Schnittstelle zwischen Controller und Browser.

Die JSP-Pages erzeugen HTML-Code, der dann an den Browser gesendet wird. Damit ist der Workflow beendet.

3.4.5 Programmhierarchie

Obwohl beim Workflow bereits alle Programmkomponenten weitgehend erläutert wurden, soll nun noch einmal näher auf die Programmhierarchie eingegangen werden. Wie schon erwähnt, ist der Programmordner /nl/strohalm/cyclos. Darin enthalten sind mehrere Packages, von denen folgende die wichtigsten sind:

Im Verzeichnis *entities* liegen die Entitäten. Dabei untergliedert sich das Verzeichnis zunächst in sehr grobe Funktionsgruppen (z.B. Accounts, Ads, Members) und dann ggf. noch einmal in feinere Gruppen. Darin enthalten sind die Entity-Klassen, die in engem Bezug zur Datenbank stehen. Mittels Hibernate findet dabei die Anbindung an die Datenbank statt. Dabei existiert für jede Entity eine entsprechende hbm.xml-Datei, die Teil des Hibernate-Frameworks ist. Die hbm.xml-Dateien liegen stets im selben Verzeichnis wie die zugehörige Entity; es gibt also kein separates Verzeichnis für die hbm.xml-Dateien. Um die Entitäten im Programm persistent zu machen, müssen die hbm.xml-Dateien noch in die Datei persistence.xml eingetragen werden, die im Verzeichnis *spring* liegt. Das Spring-Framework führt Entitäten, Serivces und DAOs zusammen und enthält einige elementare Dateien, die vor allem für die Erweiterung von Cyclos notwendig sind.

Im Verzeichnis *controls* liegen die Controller-Klassen. Auch das controls-Verzeichnis ist in Funktionsgruppen untergliedert (nach demselben Muster wie das entities-Verzeichnis). Wie schon erläutert, gibt es zu jeder elementaren Funktion eine Action-

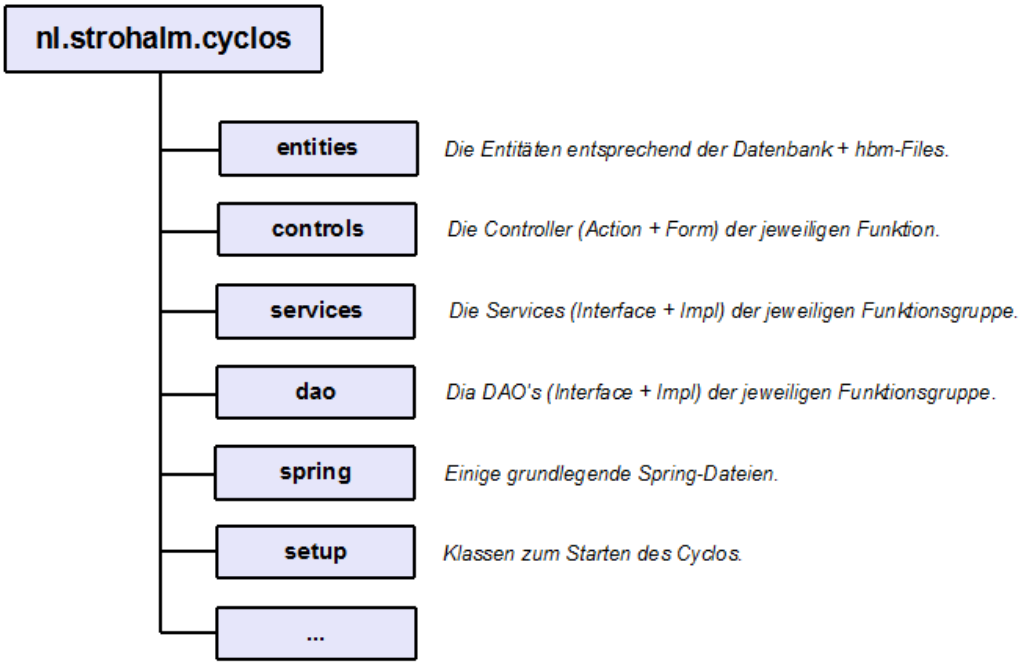

Abbildung 4: Aufbau des Programmordners (Java Klassenhierarchie).

Klasse und ggf. eine Form-Klasse.

Im Verzeichnis *services*, das ebenfalls nach selbigem Muster untergliedert ist, liegen die Service-Klassen. Zu jeder Funktionsgruppe gibt es dabei ein ServiceInterface und eine implementierende Klasse ServiceImpl. Das Serivce-Interface ist dabei eine Schnittstelle zwischen Controller und Services und ermöglicht, dass das gesamte Model (also die unterhalb der ServiceInterfaces liegenden Klassen) ausgetauscht werden kann. Damit kann Cyclos also sehr umfangreich erweitert bzw. ein gänzlich neues Model erstellt und eingesetzt werden. Die ServiceImpl-Klassen werden dabei in die Datei services.xml eingetragen, die ebenfalls im spring-Verzeichnis liegt.

Die DAO-Klassen liegen Schließlich im Verzeichnis *dao*. Die Struktur ist sehr ähnlich zu den Services; es existiert ein Interface und eine implementierende Klasse DAOImpl. Außerdem werden die DAOImpls ebenfalls zu Beans zusammengefasst und in die Datei dao.xml eingetragen, die im spring-Verzeichnis liegt.

Im Verzeichnis *setup* liegen einige grundlegende Setup-Klassen, die vor allem beim Starten von Cyclos verwendet werden. Dabei werden verschiedene Einstellungen geprüft und bei erstmaligem Ausführen die Datenbank initialisiert.

3.4.6 Erweiterung von Cyclos

Sollen im Cyclos neue Funktionen hinzugefügt werden (bzw. eine neue Funktionsgruppe), so ist entsprechend des Workflows folgendes vorzunehmen:

- Eine neue Entitätsklasse sowie eine neue hbm.xml-Datei erstellen und diese in die Datei persistence.xml eintragen.
- Für jede Funktion einen neuen Controller und ggf. eine zugehörige Form-Klasse erstellen.
- Zu der Entity ein Service-Interface sowie eine implementierende Service-Klasse erstellen und diese in die Datei services.xml eintragen.
- Zu der Entity ein DAO-Interface sowie eine implementierende DAO-Klasse erstellen und diese in die Datei dao.xml eintragen.
- Eine neue JSP-Page zu den Funktionen erstellen (einschließlich zugehöriger JS-Datei) und einen neuen Menüpunkt generieren.
- Eine neue struts-config-Datei anlegen, die entsprechenden Actions mappen und die neue Datei in die web.xml eintragen.
- Eine neue tiles-def-Datei anlegen, entsprechende definition-Tags zur Generierung der Ausgabe einfügen und die neue Datei in die Datei struts-config.xml eintragen.
- In den Application-Ressource-Dateien die Label (Texte) zu den entsprechenden Sprachen eintragen.

Natürlich muss dabei bedacht werden, dass u.U. nicht immer alle Schritte auszuführen sind. So kann es sein, dass eine neue Entität nicht benötigt wird, dass keine Ausgabe generiert wird und somit keine neue Tiles-Datei erstellt werden muss etc.

3.5 Übereinstimmung der Software mit den gewünschten Anforderungen

Aus der Untersuchung des Cyclos-Projekts hat sich ergeben, dass dieses weitgehend die grundlegenden Anforderungen eines Regionalentwicklungs-Netzwerks erfüllt und sich damit ideal zur IT-Unterstützung eignet. Das System enthält nahezu alle Funktionalitäten, die zu den allgemeinen Anforderungen solcher Netzwerke gehören. Dazu zählen vor allem das Erstellen und Suchen von Angeboten, das Verwenden von Ersatzwährungen, das Durchführen von Transaktionen und die Kooperation zwischen Mitgliedern.

Die Plattform ist relativ übersichtlich, leicht zu bedienen und enthält ein integriertes Hilfesystem. Die Daten werden sicher mit einem Datenbanksystem und Zugriff über DAO-Objekte verwaltet. Die im Kapitel 1.4 vorgestellten Anforderungen werden also weitgehend erfüllt, so dass Cyclos für den Einsatz in Regionalentwicklungs-

unternehmen grundsätzlich geeignet ist. Es wurde jedoch auch gezeigt, dass nicht alle geforderten Funktionalitäten vollständig implementiert sind, da es bspw. keine Möglichkeit gibt, mehrere Währungen zu verwenden oder Verträge abzuschließen. Die Anhalt Dessau AG könnte Cyclos somit nicht ohne eine Erweiterung bzw. Anpassung verwenden.

Die Programmarchitektur erscheint jedoch auf Grund fehlender Dokumentation und geringer Programmierunterstützung (z.B. durch Debugging-Tools, vorgefertigte Komponenten etc.) in gewisser Weise etwas problematisch. Zwar wird es Unternehmen geben, die Cyclos direkt einsetzen werden (also ohne es zu erweitern bzw. zu modifizieren), für alle anderen Unternehmen ist dies jedoch ein beachtlicher Nachteil, da die Erweiterung von Cyclos somit relativ aufwendig werden kann. Außerdem zeigt sich beim Arbeiten mit der Version 3.0.9, dass relativ häufig Exceptions bzw. allgemeine Programmfehler auftreten. Cyclos läuft also nicht sonderlich stabil, was sich jedoch mit den nachfolgenden Versionen möglicherweise ändern wird.

4 Analyse von OLAT

4.1 Einleitung

OLAT (Online Learning and Training) ist eine von der Universität Zürich seit 1999 entwickelte Software-Plattform, die eine interaktive Lernplattform (Learning Management System, LMS) für Schüler und Studierende bietet [15]. Sie ermöglicht das Bereitstellen von Lerninhalten für Schüler und Studenten, das Erstellen, Durchführen und Bewerten von Tests, Kommunikation zwischen Schülern und Studierenden sowie die Verwaltung von Kursteilnehmern. Da die Software umfangreiche und ausgereifte Funktionalitäten enthält (z.B. Chat, Wikis, Foren etc.), ist bei entsprechender Anpassung auch ein Einsatz in anderen Bereichen denkbar.

OLAT steht unter der Apache 2.0 Software Lizenz und ist somit Open Source Software [16, S. 5]. Sie kann kostenlos erworben und beliebig erweitert werden, was sicher mit dazu beigetragen hat, dass OLAT heute weitreichend eingesetzt wird. Es wurde bereits in 16 verschiedene Sprachen übersetzt und allein an der Universität Zürich läuft eine Instanz mit rund 40000 Mitgliedern [15, S. 9]. Auch heute, rund 10 Jahre nachdem die Arbeiten am OLAT begonnen wurden, wird das Programm noch intensiv weiterentwickelt. Die aktuellste Version ist die Version 6.1.1, die auch in dieser Arbeit als Grundlage verwendet werden soll.

Wie beim Cyclos soll zunächst der Funktionsumfang erläutert werden und anschließend der technische Aufbau. Der eigentliche Zweck von OLAT ist jedoch nicht der Einsatz in regionalen Entwicklungsnetzwerken, sondern an Bildungseinrichtungen (insbesondere an Hochschulen und Universitäten). Somit ist für den Einsatz in einem solchen Netzwerk die Erweiterung von OLAT unumgänglich und soll in diesem Kapitel ebenfalls erläutert werden.

4.2 Funktionsumfang

4.2.1 Mitgliederarten

Im OLAT gibt es insgesamt 6 grundlegende Mitgliederarten, die als *OLAT-Systemrollen* bezeichnet werden [17]. Diese sind:

- Gäste
- Benutzer
- Autoren
- Gruppenverwalter
- Benutzerverwalter
- Systemadministratoren

Gäste besitzen keinen eigenen Account. Sie sind nur Besucher der Website und können ggf. auf Lerninhalte zugreifen, sofern diese für Gäste freigegeben wurden.

Benutzer werden i.A. den größten Anteil an allen registrierten Usern bilden. Sie besitzen einen eigenen OLAT-Account mit einem eindeutigen, nicht-änderbaren Usernamen, einem persönlichen Bereich (u.a. Profil) und einer bestimmten Menge an Speicherplatz, den sie frei nutzen können. Sie können auf allgemeine Lerninhalte zugreifen, sofern diese für sie freigegeben sind.

Autoren sind Benutzer, die zusätzlich Lerninhalte erstellen bzw. in das System importieren können. Sie können die Lerninhalte beliebig anpassen, Zugriffsrechte vergeben, Kursteilnehmer eines Kurses verwalten etc.

Der Systemadministrator hat zusätzliche Rechte zum Konfigurieren des Systems und Verwalten der Mitglieder.

Zudem gibt es noch die beiden Rollen Gruppenverwalter und Benutzerverwalter. Diese sind relativ selten; sie decken jeweils einen Teil der Rechte des Admins ab. Sie stehen also hierarchisch zwischen Autoren und Systemadministratoren.

Die Rolle eines Mitglieds kann im OLAT auch erweitert werden. So kann z.B. einem Benutzer erlaubt werden, bei der Verwaltung eines Kurses zu helfen, so dass er also auch Rechte ausübt, die eigentlich nur Autoren zustehen.

4.2.2 Grundlegender Aufbau der Website

Die OLAT-Website besteht aus einem Kopfbereich (Top Navigation Tool), in dem allgemeine Funktionen zur Verfügung stehen und einem Hauptbereich, der aus meh-

reren Tabs besteht.[21] Es gibt statische Tabs und dynamische Tabs. Benutzer verfügen i.A. über 3 statische Tabs: Home, Gruppen, Lernressourcen. Jedes statische Tab enthält auf der linken Seite eine Menüleiste (Content Menu), so dass auf die verschiedenen Funktionen zugegriffen werden kann. Rechts daneben befindet sich dann der Ausgabebereich (Content Area), der den größten Teil der Seite einnimmt. Am rechten Rand des Content Areas befindet sich in einer schmalen Spalte die Toolbox, in der, wenn nötig, auf einige Werkzeuge zugegriffen werden kann (z.B. Erstellen einer Arbeitsgruppe etc.). Die Toolbox kann ggf. auch ausgeblendet sein.

Einige Inhalte werden in neuen Tabs angezeigt, z.B. die Volltextsuche oder der Zugriff auf ein Benutzerprofil oder eine Gruppe. Hierbei handelt es sich um dynamische Tabs, die der Benutzer zu jeder Zeit wieder schließen kann. Es können standardmäßig bis zu 5 dynamische Tabs geöffnet werden, die rechts neben den statischen Tabs erscheinen. Anschließend wird eine Fehlermeldung ausgegeben, die besagt, dass zu viele Tabs geöffnet sind. Der Benutzer muss somit ein Tab schließen, bevor er ein neues öffnet.

4.2.3 Funktionen

In diesem Abschnitt sollen die Funktionen dargelegt werden, die Benutzer und Gäste besitzen. Dazu soll zunächst das grundlegende Funktionsprinzip erläutert werden, bevor dann auf die einzelnen Funktionen näher Bezug genommen wird.

Grundlegendes Funktionsprinzip

Lernressourcen (Lerninhalte) sind die elementaren Bestandteile von OLAT [17, S. 8]. Eine Lernressource kann beispielsweise ein Fragebogen oder Test sein, ein Dokument oder ein Kurs. Die Gesamtmenge der Lernressourcen heißt auch wieder Lernressourcen (oder Ablage, Repository). Lernressourcen können nur von Autoren (bzw. höheren Rollen) erstellt werden. Jede Lernressource besitzt Meta-Daten, z.B. Name, Beschreibung, Typ, Autor etc. sowie Sichtbarkeits- und Zugriffsrechte, die festlegen, wer die Lernressource sehen und darauf zugreifen darf. Der Autor und Verwalter der Lernressource ist entsprechend für die Vergabe von Zugriffsrechten und das Erstellen der Meta-Daten verantwortlich. Eine weitere Besonderheit von Lernressourcen ist, dass diese beliebig importiert und exportiert werden können; ein Kurs kann beispielsweise von einer OLAT-Instanz auf eine andere exportiert werden.

Kurse sind spezielle Lernressourcen, die direkter Bestandteil des Softwaresystems sind und somit einen Dokumenttyp darstellen, der direkt von dem OLAT-Team entwickelt wurde [15, S. 23] [17, S. 10 f.]. Kurse repräsentieren Vorlesungen, (Lern- etc.) Kurse, Seminare, allgemeine Veranstaltungen etc., in die sich Mitglieder einschreiben können. Die Kurse können hierbei wiederum zahlreiche Lernressourcen beinhalten, die im OLAT als *Kursbausteine* bezeichnet werden. Kurse können damit unterschiedlich komplex sein. Manche Kurse werden womöglich nur aus ein paar

[21] http://www.olat.org/docu/dev/olat_dev_docu_one_page.html#interaction_design

wenigen HTML-Seiten bestehen (dann haben die Kurse eher informativen Charakter), manche Kurse werden womöglich sehr komplex sein und bspw. ein Ordnersystem, ein Forum, ein Wiki und viele andere Komponenten enthalten (dann haben sie kooperativen Charakter). Damit zeigt sich bereits das kooperative Potential, das solche Kurse bieten - Kommunikation wird im OLAT weitreichend unterstützt (der Autor legt jedoch fest, wie viel Kommunikation möglich ist, also wie viele kooperative Kursbausteine er in dem Kurs freischaltet).

Wird OLAT bspw. an einer Hochschule oder Universität eingesetzt, so wäre es sinnvoll, für jede Vorlesung bzw. jedes Seminar einen eigenen Kurs zu erstellen. In dem Kurs können sich Studenten dann eintragen und über Foren und Wikis miteinander kommunizieren, es können Termine, Vorlesungsskripte etc. veröffentlicht und Fragebögen, Tests etc. für die Kursteilnehmer erstellt werden. Zudem haben Dozenten und Hochschulpersonal einen Überblick, welche Studenten in welchen Vorlesungen und Seminaren eingetragen sind - OLAT bietet sich somit auch direkt für die Verwaltung der Schüler und Studenten einer Schule, Hochschule oder Universität an.

Im folgenden sollen einige wesentliche Kursbausteine vorgestellt werden, die ein OLAT-Kurs enthalten kann. Die Auflistung zeigt, wie komplex Kurse sein können und welche Möglichkeiten damit letztendlich zur Verfügung stehen:

- Einzelne, einfache HTML-Seiten (z.B. eine Willkommen-Seite etc.)
- Ordnersystem, in dem beliebige Dateien abgelegt werden können
- HTML-formatierte Lerninhalte
- Tests, Fragebögen, Aufgaben
- Bewertungen von Tests
- Forum
- Wiki
- Glossar

Da Kurse Lernresourcen sind, verfügen sie über entsprechende Sichtbarkeits- und Zugriffsbeschränkungen. OLAT bietet hierbei auch dynamische Zugriffsbeschränkungen, z.B. Zugriff nur innerhalb eines bestimmten Zeitraums oder in Abhängigkeit von in Tests erreichten Ergebnissen [17, S. 11].

Zu jedem Kurs können mehrere Lerngruppen existieren. In *Lerngruppen* können Mitglieder des Kurses eingetragen sein und haben dann die Möglichkeit, in einem kleineren Bereich miteinander zu kommunizieren. Jede Lerngruppe ist dabei genau einem Kurs zugeordnen und kann ebenfalls wieder Zugriffsbeschränkungen besitzen, die vom Ersteller der Lerngruppe festgelegt werden. Außerdem kann der Autor für jede Lerngruppe eine Obergrenze an Teilnehmern festlegen. OLAT bietet hierbei auch die Möglichkeit einer Warteliste, so dass Wartende automatisch in die Lerngruppe eingetragen werden, falls ein bereits eingetragenes Mitglied aus der Gruppe austritt. Mehrere Lerngruppen mit ähnlichen Funktionen können zu *Lernbereichen* zusammengefasst werden [15, S. 19].

Die Verwendung von Kursen, Lernbereichen und Lerngruppen zeigt, dass OLAT ein beachtliches didaktisches Konzept aufweist. So können bsp. in einem Kurs mehrere

100 Personen eingeschrieben sein, zu dem mehrere Dutzend Lerngruppen existieren können, die zu ein paar wenigen Lernbereichen zusammengefasst sind.

Neben den Lerngruppen gibt es noch Rechtegruppen und Arbeitsgruppen. *Rechtegruppen* ermöglichen ausgewählten Personen, auf bestimmte Werkzeuge eines Kurses zugreifen zu können. Sie werden vom Kursautor verwaltet. *Arbeitsgruppen* sind hingegen völlig unabhängig von Kursen; sie existieren autonom und können von jedem Benutzer erstellt werden. Die Benutzer, die sie anlegen, verwalten diese und können beliebig viele OLAT-Benutzer in die Gruppe einladen (Nicht-Mitglieder haben keinen Zugriff auf die Gruppe). Arbeitsgruppen sind somit eher für das Selbststudium oder für Projekte, die unabhängig von Kursen sind, konzipiert [15, S. 17 f., S. 21 f.].

Allgemeine Funktionen

Oberhalb des eigentlichen Menüsystems hat der Benutzer Zugriff auf allgemeine Funktionen. So sieht er, wie viele Personen neben ihm noch eingeloggt sind, er kann den Instant-Messenger öffnen und seinen Chat-Status ändern (z.B. verfügbar, abwesend, nicht stören etc.), er kann die OLAT-Hilfe öffnen, er hat die Möglichkeit, die aktuelle Seite zu drucken, und er kann sich ausloggen.

Besonders hervorzuheben ist die Volltextsuche. Hierbei gibt der Benutzer ein Stichwort ein und OLAT durchsucht im Anschluss alle Lernressourcen (z.B. Foren, Wikis, PDF- und Word-Dokumente etc.) nach diesem Begriff und gibt die gefundenen Resultate aus. Natürlich werden dem Benutzer dabei nur die Dokumente angezeigt, auf die er auch zugreifen darf [15, S. 15]. Da innerhalb einer OLAT-Instanz mehrere tausend Dokumente enthalten sein können, gibt es auch eine erweiterte Volltextsuche, wo bspw. der Autor des Dokuments, Titel, Beschreibung, Erstelldatum, Bereich (Wiki, Forum etc.) mit bei der Suche berücksichtigt werden können. Damit bietet OLAT ein leistungsfähiges Mittel zum schnellen und gezielten Suchen von Dokumenten bzw. Lerninhalten.

Persönlicher Bereich

Jeder Benutzer besitzt einen Home-Bereich, in dem er persönliche Einstellungen vornehmen, seinen Account verwalten und grundlegende bzw. persönliche Funktionen ausführen kann. Auf der Startseite finden sich dabei zahlreiche Links zu elementaren OLAT-Funktionen, z.B. Links zu den Kursen, eine Übersicht über alle Gruppen, in die der Benutzer eingetragen ist, neue Benachrichtigungen, allgemeine Informationen zu OLAT etc.

Unter *Einstellungen* kann der Benutzer Einstellungen zu seinem Profil, zu seinem Account und zum Instant-Message-System vornehmen. Das Profil enthält zahlreiche Angaben, wobei jedoch nur Benutzername, Vorname, Nachname und E-Mail verbindlich sind. Alle anderen Angaben (z.B. Adresse, Geburtsdatum, Matrikelnummer etc.) sind optional. OLAT ermöglicht auch das Hochladen eines Fotos (Größe bis 500 KB) und das Verfassen eines persönlichen Textes. In dem Menü *Visitenkarte* kann der Benutzer einstellen, welche Angaben für andere Mitglieder sichtbar sind. Er allein entscheidet also, wie viele persönliche Daten er preisgibt.

Jeder Benutzer besitzt einen *persönlichen Ordner* mit einer bestimmten Menge an verfügbarem Speicherplatz. Der Ordner untergliedert sich in die Verzeichnisse *private* und *public*. Im privaten Verzeichnis gespeicherte Dokumente können nur von dem Benutzer selbst eingesehen werden. Auf Dokumente, die im öffentlichen Verzeichnis gespeicherte sind, können hingegen auch andere Benutzer zugreifen [15, S. 14].

Im Menü *Andere Benutzer* kann nach anderen Mitgliedern gesucht werden. Hierzu muss entweder ein Benutzername, ein Vor- oder Nachname oder eine E-Mail-Adresse eingegeben werden. Liegen Resultate vor, so werden diese in einer Liste ausgegeben. Standardmäßig wird die Liste dabei nach Benutzername sortiert, jeder Benutzer kann die Sortierreihenfolge aber auch beliebig ändern (z.B. nach Name oder E-Mail-Adresse).

OLAT bietet für verschiedene Aktionen *Benachrichtigungen*, die automatisch vom System generiert werden und im Menü *Benachrichtigungen* bzw. auf der Home-Seite gelesen werden können. Solche Benachrichtigungen können z.B. erfolgen, nachdem auf einen Forum-Beitrag geantwortet wurde, eine Seite im Wiki geändert wurde etc. Der Benutzer kann entscheiden, für welche Aktionen er Benachrichtigungen empfangen möchte und ob er auch per E-Mail informiert werden möchte [15, S. 14].

Im Gegensatz zu vielen anderen Web-Plattformen gibt es im OLAT kein vollständiges, internes Nachrichtensystem. Möchte man mit anderen Benutzern direkt in Verbindung treten, kann man i.A. nur eine E-Mail an das Mitglied senden. Allerdings bietet OLAT in den Kursen und Gruppen eine Vielzahl weiterer Kommunikationsmöglichkeiten wie Forum, Wiki oder Chat sowie den Instant-Message-Service. Insofern erscheint die Bedeutung eines internes Nachrichtensystems nicht besonders groß.

In jedem Kurs können Benutzer persönliche Notizblätter erstellen. Im Menü *Notizblätter* werden diese verwaltet.

Gruppen

Im Tab *Gruppen* werden alle Gruppen aufgelistet, in denen man eingetragen ist. Es wurde bereits erwähnt, dass es im OLAT drei Arten von Gruppen gibt: Lerngruppen als zentraler Bestandteil eines OLAT-Kurses, Arbeitsgruppen, die unabhängig von Kursen erstellt werden, sowie Rechtegruppen, die einigen Mitgliedern erweiterte (Zugriffs-) Rechte gewähren.

In dem Tab können entweder alle Gruppen anzeigen werden, in denen man eingetragen ist, oder separat alle Arbeitsgruppen, Rechtegruppen und Lerngruppen. Klickt man eine Gruppe an, so öffnet sich diese in einem neuen Tab und man kann auf entsprechende Werkzeuge der Gruppe zugreifen.

Arbeitsgruppen werden von einem Mitglied erstellt, welches dann der Besitzer der Gruppe ist. Dieser kann auch weitere Mitglieder als Gruppenbesitzer ernennen, so dass Arbeitsgruppen auch von mehreren Personen verwaltet werden können. Zudem

können beliebig viele Mitglieder in die Gruppe aufgenommen werden. Die Besitzer legen hierbei fest, ob die Mitglieder sich untereinander sehen können oder nicht. Nach Erstellen der Arbeitsgruppe können dieser dann beliebig viele Werkzeuge hinzugefügt werden. Einige Werkzeuge sind z.B. Kalender, Mitgliederliste, Chat, Wiki, Forum, E-Mail und ein Ordnerverzeichnis zum Ablegen von Dateien. Dabei kann ein Werkzeug niemals mehrfach hinzugefügt werden (es kann also keine 2 Wikis geben etc.) [15, S. 17 ff.].

Lerngruppen unterscheiden sich von Arbeitsgruppen dadurch, dass sie an einen Kurs gebunden sind. Allerdings gibt es auch viele Gemeinsamkeiten mit den Arbeitsgruppen. So gibt es ebenfalls einen Besitzer der Gruppe, der diese verwaltet und Mitglieder hinzufügen kann, und es ist ebenfalls möglich, beliebige Werkzeuge der Gruppe hinzuzufügen.

Für die Einschreibung in Lerngruppen gibt es zwei Möglichkeiten: Entweder Mitglieder tragen sich selbstständig in eine Gruppe ein, oder sie werden vom Kursautor oder einem Betreuer eingetragen. Der erste Fall eignet sich z.B. bei Gruppenarbeiten, wo jeder Teilnehmer sich individuell in eine beliebige Gruppe eintragen soll. Dabei müssen die künftigen Mitglieder selbstständig den Kurs starten, in dem die gewünschte Lerngruppe enthalten ist und sich über das Menü *Einschreiben* in die gewünschte Lerngruppe einschreiben. Im zweiten Fall muss der Kursautor bzw. Betreuer die Mitglieder selbst in entsprechende Gruppen eintragen, was nur möglich ist, wenn er weiß, in welche Gruppe jedes Mitglied eingetragen werden muss. Neuen Mitgliedern einer Gruppe können hierbei zwei Rollen zugewiesen werden: Betreuer oder Teilnehmer. Teilnehmer sind gewöhnliche Mitglieder, die lediglich das Recht haben, auf die Gruppenwerkzeuge und Funktionalitäten zuzugreifen. Betreuer haben zusätzlich das Recht, neue Mitglieder in die Gruppe einzuladen. Sie unterstützen dem Kursautor bei seiner Arbeit, ihre Systemrolle ist aber nach wie vor *Benutzer*.

Das Austragen aus einer Gruppe funktioniert ähnlich wie das Eintragen. Allerdings ist es dem Kursautor auch möglich, das Austragen aus der Gruppe zu verbieten [15, S. 19].

Mitglieder können einzelnen Rechtegruppen hinzugefügt werden, die ebenfalls von einem Kurs abhängig sind bzw. sich auf genau einen Kurs beziehen. Sie eignen sich zum Beispiel dafür, einigen Mitgliedern mehr Rechte zu gewähren als Standard-Mitgliedern, ohne ihnen aber gleich das Kursautor-Recht gewähren zu müssen. Es können also Rechtegruppen erstellt werden, deren Mitglieder mehr Rechte als Standardbenutzer haben, aber weniger als Kursautoren.

Rechtegruppen werden vom Kursautor angelegt und ggf. mit Werkzeugen ausgestattet. Der Kursautor legt dabei die konkreten Rechte fest, die ein Mitglied dieser Gruppe besitzt. Das kann z.B. das Recht zur Archivierung von Daten (z.B. Forumseinträgen) oder die Gruppenverwaltung sein. Die Mitglieder einer Rechtegruppe werden vom Kursautor eintragen und ggf. auch wieder ausgetragen; selbst austragen können sich die Mitglieder nicht [15, S. 21 f.].

Lernressourcen

Im dritten Tab hat man Zugriff auf die Lernressourcen, die im OLAT angelegt wurden. Wie bereits erwähnt, sind Lernressourcen die zentralen Bestandteile von OLAT und können z.B. Wikis, Tests, Fragebögen oder Kurse sein. Letztere haben einen besonders hohen Stellenwert im OLAT.

Im Lernressourcen-Tab gibt es drei Menüs: Katalog, Suchmaske und Kurse. Im *Katalog* hat man Zugriff auf alle Lernressourcen. Da es u.U. sehr viele Lernressourcen auf einer OLAT-Plattform geben kann, kann ein Ordnersystem zur hierarchischen Speicherung der Lernressourcen verwendet werden. Somit wird dem Benutzer ein Überblick über alle Lernressourcen ermöglicht sowie ein relativ schnelles Auffinden bestimmter Lernressourcen.

Die *Suchmaske* ermöglicht eine gezielte Suche nach Lernressourcen. Hierbei kann nach einem Titel, einem Autor, einer Beschreibung oder einem bestimmten Typ (Art der Lernressource) gesucht werden. Liegen Resultate vor, so werden diese wieder in einer Liste ausgegeben, von wo aus man auf die entsprechende Ressource zugreifen kann (es werden stets nur die Lernressourcen als Resultat angezeigt, auf die man Zugriff hat) [15, S. 25].

Im Menü *Kurse* findet schließlich eine alphabetische Auflistung aller Kurse statt, auf die man Zugriff hat. Durch Anklicken eines Kurses öffnet sich dieser in einem neuen Tab.

Autoren haben in dem Tab Lernressourcen noch einige weitere Rechte. Sie können auch gezielt nach Lerninhalten suchen, Lernressourcen einer anderen OLAT-Instanz importieren und neue Lernressourcen erstellen. Zum Erstellen von Kursen, Fragebögen und Tests sind im OLAT bereits entsprechende Tools (Editoren) enthalten, so dass diese relativ einfach generiert werden können [15, S. 26].

4.3 Technischer Aufbau

4.3.1 Einführung

Im Gegensatz zu Cyclos ist OLAT deutlich komplexer und umfasst deutlich mehr Programmcode. Der technische Aufbau soll daher nicht ganz so ausführlich betrachtet werden.

In diesem Abschnitt werden zunächst grundlegende Aspekte sowie die Verzeichnisstruktur erläutert, die Programm-Architektur sowie die Erweiterungsmöglichkeiten. Im nächsten Abschnitt wird dann erläutert, inwiefern OLAT erweitert werden muss,

um in Regionalentwicklungsunternehmen eingesetzt werden zu können.

4.3.2 Verzeichnisstruktur

Das OLAT-Programm, das direkt von der OLAT-Website ausgecheckt werden kann, heißt *olat3* und beinhaltet mehrere Verzeichnisse, u.a. *database*, *doc*, *conf* und *webapps*.[22]

In dem Verzeichnis *database* liegen grundlegende Konfigurationsdateien zum Erstellen der Datenbank. Die Datei setupDatabase.sql dient dabei zur Erzeugung der Datenbank; sonst befinden sich auch noch einige Dateien zum Upgraden der Datenbank von einer älteren auf eine neuere OLAT-Version in dem Verzeichnis. Im Verzeichnis *doc* liegt die OLAT-Dokumentation, im Verzeichnis *conf* liegen grundlegende Konfigurationsdateien, z.B. Dateien für den Apache Server.

Außerdem befinden sich in dem olat3-Verzeichnis die beiden Dateien build.properties und build.xml, in denen grundlegende Parameter zur Ausführung bzw. zum Kompilieren von OLAT enthalten sind.

Entscheidend für die eigentliche Programmausführung ist jedoch der *webapps*-Ordner. Darin enthalten sind die Verzeichnisse help, examples, static und WEB-INF.

Im Verzeichnis *help* befindet sich die gesamte OLAT-Hilfe, im Verzeichnis *examples* befinden sich einige Programmbeispiele und Demos und im Verzeichnis *static* befinden sich die statischen Dateien, also z.B. Bilder, Icons und theme-Dateien.

Im *WEB-INF*-Verzeichnis liegen schließlich die eigentlichen Programmdateien von OLAT; wie üblich ist der WEB-INF-Ordner vom Client nicht sichtbar.

Der WEB-INF-Ordner untergliedert sich noch einmal in die Verzeichnisse classes, lib, src, patchesSrc und test. Im *classes*-Verzeichnis liegen die gesamten kompilierten Java-Klassen (Bytecode). Im *lib*-Verzeichnis liegen die Bibliotheken und Lizenzen aller verwendeten Frameworks (z.B. Hibernate, Spring, MySQL etc.). Im Verzeichnis *src* sind die Java-Quellcode-Dateien enthalten, auf die später noch näher eingegangen wird. Im Verzeichnis *patchesSrc* liegen Quellcode-Dateien geänderter Bibliotheken und im Verzeichnis *test* befinden sich einige Testdaten.

4.3.3 Quellcode

Das Klassenverzeichnis hat den Namen org.olat und enthält darin zahlreiche Packages, in denen die Quellcode-Dateien liegen. Etwas gesondert betrachtet werden muss

[22] http://pcai042.informatik.uni-leipzig.de/swp/SWP-08/swp08-7/Recherchebericht.pdf

der Programmkern von Olat (OLAT Core), der im Package org.olat.core enthalten ist. Er umfasst alle grundlegenden Klassen zur Programmausführung, u.a. auch das von den Programmiereren entwickelte GUI-Framework. Wird das OLAT Projekt standardmäßig ausgecheckt, so ist der OLAT Core darin nicht enthalten.

Der Quellcode-Ordner ist etwa 14,7 MB groß (ohne OLAT Core) und enthält rund 1.423 Java-Klassen. Zusätzlich enthalten sind die Hibernate-Dateien (hbm.xml-Dateien), die sich in demselben Verzeichnis befinden wie die zugehörige Java-Klasse und denselben Namen tragen. Auch einige Konfiguratonsdateien (properties) und HTML-Dateien sind in dem Programmordner enthalten. In einigen Packages befindet sich auch ein Ordner _i18n, in dem sich die Labeltexte für alle GUI-Komponenten und Messages befinden. Dabei gibt es stets zwei Dateien: LocalStrings_de.properties für die deutsche Sprache und LocalStrings_en.properties für die englische Sprache. Hiermit zeigt sich, dass die Texte also nicht in einer zentralen Datei gespeichert sind, sondern lokal in den einzelnen Packages.

Das Klassenverzeichnis weist eine relativ flache Hierarchie auf. Das zentrale OLAT-Package (org.olat) gliedert sich in rund 35 weitere Packages, wobei manche grundlegende Programmfunktionen umfassen (z.B. admin, user, gui, test etc.), während andere bereits konkrete Funktionsgruppen bzw. Programmbausteine repräsentieren (z.B. course, login, group etc.). Die Packages enthalten oft weitere Packages, die somit die 3. Ebene in der Hierarchie bilden. Obwohl es bis zu 5 Ebenen geben kann, liegen fast alle Java-Klassen auf 2. oder 3. Ebene; in den tieferen Ebenen liegen dann meist nur die Konfigurations- und HTML-Dateien.

Der Quellcode entspricht etwa den allgemeinen Java-Konventionen. So werden sprechende Namen für Bezeichner verwendet und die Klassenhierarchie sinnvoll gegliedert. Allerdings erscheint die Strukturierung des Quellcodes oft unübersichtlich, da auch bei umfangreichen Methoden kaum Leerzeilen zur Strukturierung verwendet werden und die Zeilenlänge von 80 Zeichen oftmals deutlich überschritten wird.

OLAT ist weitgehend gut dokumentiert. Die Programmierer haben sehr viel Wert auf die JavaDOC gelegt, so dass eine einigermaßen vollständige JavaDoc vorliegt. Einige Klassen und Methoden sind trotzdem nur unvollständig dokumentiert, wobei es sich dann meist um weniger wichtige Klassen handelt, deren Bedeutung aus dem Kontext relativ gut hervorgeht. Auch wurden an vielen Stellen Inline-Kommentare verwendet, welche die teilweise komplexen Methoden zusätzlich beschreiben.

Zusätzlich zu der JavaDoc und den allgemeinen Kommentaren bietet OLAT zahlreiche weitere Spezifikationen an, darunter auch eine technische Dokumentation, eine Funktionsübersicht, ein Benutzerhandbuch u.v.m. Auf Grund der Bedeutung der Software wurden auch an vielen Instituten und Hochschulen Arbeiten und Analyseberichte zum OLAT verfasst, u.a. auch an der Universität Leipzig.

Mit der JavaDoc und den Programmspezifikationen ist somit eine Einarbeitung in das OLAT trotz der relative hohen Komplexität der Software ziemlich gut möglich. Im OLAT integrierte Debug-Tools, z.B. der GUI Debug Mode, unterstützen das Entwickeln bzw. Erweitern der Software ebenfalls erheblich.

4.3.4 Rechte und Rollen im OLAT

Neben den bereits vorgestellten Systemrollen (Gast, Mitglied, Autor etc.) gibt es im OLAT drei Business Groups, die ebenfalls bereits vorgestellt wurden: Lerngruppen, Rechtegruppen und Arbeitsgruppen. In den Gruppen sind entsprechender Mitglieder (Identities) enthalten. Zu jeder Gruppe gibt es mindestens eine SecurityGroup, wobei es grundsätzlich zwei Arten von SecurityGroups gibt: Owners und Participants. Jedes Mitglied der Gruppe ist dabei genau einer der beiden SecurityGroups zugeordnet.

Gehört das Mitglied der SecurityGroup Owner an, so heißt dies, dass es mit zu den Besitzern der Gruppe gehört und somit Änderungen an der Gruppe vornehmen kann. Die meisten Mitglieder der Gruppe werden jedoch in der Gruppe Participants eingetragen sein, so dass sie kein Änderungsrecht besitzen, aber allgemein an der Gruppe teilhaben dürfen. Bei Arbeitsgruppen gibt es die SecurityGroup Owner nicht; hier sind alle Teilnehmer Participants.

Benutzer haben entsprechend ihrer Rolle bzw. entsprechend der Business Groups, in denen sie eingetragen sind, verschiedene Rechte. Die Rechte sind positiv und additiv. Das heißt, dass im OLAT Rechte immer ausdrücken, was ein User ausführen darf. Additiv bedeutet, dass die Rechte des Users der Gesamtheit aller einzelnen Rechte entspricht, die der User besitzt [18, S. 15].

4.3.5 Programmarchitektur

Wie nahezu jede andere Software-Plattform besteht auch das OLAT aus einer Schichtenarchitektur, die in etwa den allgemeinen Konventionen folgt, wie sie bereits im 3. Kapitel vorgestellt wurden. Insgesamt besteht das OLAT dabei jedoch aus 7 Schichten (Abbildung 5).

Die oberste Schicht stellt den Servlet Container dar, also z.B. den Tomcat. Dies ist die Schnittstelle zum Anwender der Software, das heißt, über diese Schicht werden Aktionen vom Benutzer ausgelöst und müssen entsprechende Ausgaben generiert werden.

OLATServlet und Dispatcher

Das OLAT-Servlet ist eine Art Listener, der darauf wartet, dass der Benutzer eine bestimmte Aktion ausführt. Dabei werden mehrere Dispatcher verwendet, um den HTTP-Request weiter zu verarbeiten. Die Klasse *DispatcherAction* kann dabei als der zentraler Dispatcher betrachtet werden. Sobald ein Request ausgelöst wird, wird diese Klasse aufgerufen, die anschließend den vorderen Teil des Requests analysiert und daraus eindeutig bestimmen kann, welcher untergeordnete Dispatcher für die weitere Verarbeitung des Requests verantwortlich ist und aufgerufen werden muss.

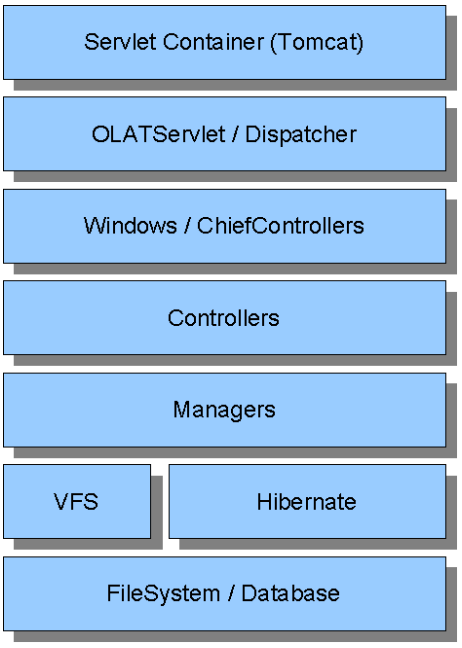

Abbildung 5: OLAT: Application Layers, http://www.olat.org/docu/dev/olat_dev_docu_one_page.html#ApplicationLayers

Es gibt im OLAT mehrere Arten von Dispatchern:

- DMZDispatcher
- AuthenticalDispatcher
- ShibbolethDispatcher

Der *DMZDispatcher* verarbeitet alle Aktionen, die von noch nicht authentifizierten Mitgliedern ausgelöst wurden. Der *AuthenticalDispatcher* behandelt Aktionen von bereits authentifizierten Mitgliedern. Der *ShibbolethDispatcher* bearbeitet schließlich asynchron ausgelöste Requests des IdentityProviders.

Neben dem OlatServlet existieren noch eine Reihe weiterer Servlets, z.B. das StaticsServlet, das Requests statischer Objekte (z.B. Bilder) verarbeitet.[23]

ChiefControllers und Windows

Ein Window ist die serverseitige Darstellung eines Browser-Fensters und besitzt ähnlich wie in Swing ein Contentpane, in dem verschiedene Komponenten aufgenommen werden können. Jedes Window ist dabei einem ChiefController zugeordnet, der die Navigation in dem Fenster steuert. Eine UserSession, die jedem Benutzer zugeordnet ist, speichert dabei die Windows des Users.[24]

[23] http://www.olat.org/docu/dev/olat_dev_docu_one_page.html#d0e2890
[24] http://www.olat.org/docu/dev/olat_dev_docu_one_page.html#d0e2949

Controller

Die Controller werden bei einer vom Benutzer ausgelösten Aktion aufgerufen und besitzen eine Initialkomponente, die als visuelle Repräsentation des Controllers dient und später in ein HTML-Fragment gerendert wird, das Teil der gesamten Website ist.[25]

Das View ist ebenfalls Teil der Controller-Ebene und steht den Controllern sehr nahe. View-Klassen, die für das Erstellen der Ausgabe verantwortlich sind, heißen ebenfalls Controller oder Sites (z.B. GroupsSite). Die meisten Objekte, die sie verwenden, z.B. MainLayoutController, WindowsControl, UserRequest, die gesamten Grafikkomponenten etc., sind Teil des OLAT Cores und liegen in dem umfangreichen Package org.olat.core. bzw. org.olat.core.gui. Sie müssen daher vom Entwickler in den meisten Fällen nicht geändert bzw. erweitert werden. Damit wird das Erstellen des Views enorm vereinfacht. Soll beispielsweise ein Formular angelegt werden, kann mit der Methode *addFormElement()* auf einfache Weise eine neue Komponente zu einem Formular hinzugefügt werden. Die eigentliche Umsetzung in HTML (Rendering) geschieht dann intern und ist für den Entwickler nicht weiter wichtig.

Das Erstellen der Ausgabe ist somit relativ einfach; der Programmierer muss keine JavaScript- oder JSP-Dateien erstellen wie bei anderen Programmen, sondern verwendet das vom OLAT-Team entwickelte Brasato-Framework.[26] Dieses Framework orientiert sich dabei stark an dem SWT- und Swing-Framework von Java und stellt Container zur Verfügung, in die weitere Container oder verschiedene GUI-Komponenten eingefügt werden können.[27]

Manager

Die Manager können als das Model des Programms angesehen werden und besitzen grundlegende Funktionen bezüglich der Geschäftslogik. Auf Grund der MVC-Architektur sind Funktionen wie *read*, *write*, *update*, *delete*, *find*, *load* etc. somit in den Manager-Klassen enthalten und nicht in den Controllern. In den Managern kann direkt mit SQL-Anfragen auf die Datenbank zugegriffen werden, um Daten einzufügen, zu ändern, zu löschen oder auszugeben. Manager bestehen üblicherweise aus einem Interface und einer implementierenden Klasse.

VFS

Das im OLAT implementierte VFS (Virtual File System) stellt eine Baumstruktur an Ordnern und Dateien dar, die im OLAT verwendet werden. In dieser Schicht wird also von dem eigentlichen Dateisystem abstrahiert. Oft existiert zu den einzelnen Objekten (Datei oder Ordner) ein *SecurityCallback*, der festlegt, welche Aktionen auf das Objekt erlaubt sind (z.B. lesen, schreiben, löschen); existiert dieser nicht, so sind standardmäßig alle Operationen auf das Objekt erlaubt. Die Operationen werden dabei innerhalb des Baums von einem Knotenpunkt bis zu den Blättern vererbt. Darf ein Ordner z.B. nicht gelöscht werden, so dürfen auch alle Unterordner

[25] http://www.olat.org/docu/dev/olat_dev_docu_one_page.html#d0e2966
[26] http://www.olat.org/docu/dev/olat_dev_docu_one_page.html#d0e2485
[27] http://www.olat.org/docu/gui/ch01s01.html

und darin enthaltene Dateien nicht gelöscht werden.[28]

Datenbank und Hibernate

Daten aus der Datenbank werden in Entitätsklassen gespeichert, die über eine hbm.xml-Datei mit dem Hibernate persistent gemacht werden. Solche Klassen besitzen, wie üblich, eine Reihe von Attributen (i.A. genau die Attribute der Datenbank-Tabelle, die sie repräsentieren) mit entsprechenden Gettern und Settern. Sie sind in OLAT stets von *PersistentObject* abgeleitet, was garantiert, dass jede der Klassen die Attribute key, creationDate und lastModified besitzt. Die persistenten Klassen befinden sich üblicherweise im selben Verzeichnis wie die zugehörigen Manager und besitzen ein Interface und eine implementierende Klasse. Auch befinden sich darin die hbm.xml-Dateien zur Anbindung an das Hibernate.

4.3.6 OLAT erweitern

Um OLAT zu erweitern, erstellt man ein neues Package, in dem alle neuen Klassen bzw. Packages enthalten sind. Danach wird das Package zu einem jar zusammengefasst und in das Verzeichnis WEB-INF/lib eingefügt. Damit das jar vom OLAT erkannt wird, muss es noch registriert werden. Dies geschieht in der Datei olat_extensions.xml, wo eine neue Zeile eingefügt wird, die auf das Package in dem lib-Ordner verweist. In dem Package können dann auch statische Daten, Konfigurationsdateien etc. enthalten sein.

Eine andere Variante der Erweiterung ist es, bestehende Klassen einfach zu ändern, zu erweitern bzw. weitere Klassen in das bestehende Klassenverzeichnis einzufügen. Diese Lösung erscheint allerdings weniger elegant, weil eventuelle Fehler dann schwerer aufzufinden sind bzw. deutlich größeren Einfluss auf andere Programmteile haben könnten, als in der ersten Variante.

Eine ausführliche Anleitung zum Erstellen eines neuen Workflows in OLAT ist in[29] beschrieben.

[28] http://www.olat.org/docu/dev/olat_dev_docu_one_page.html#vfs
[29] http://www.olat.org/docu/dev/olat_dev_docu_one_page.html#d0e3236

4.4 Verwendung von OLAT in Regionalentwicklungs-Netzwerken

4.4.1 Probleme und Grenzen von OLAT

In diesem Abschnitt soll erläutert werden, inwiefern man OLAT anpassen bzw. erweitern muss, damit es in regionalen Entwicklungsnetzwerken verwendet werden kann, so wie sie im Kapitel 1 vorgestellt wurden. Das OLAT bietet zwar eine Vielzahl von Funktionalitäten, ist aber als reine Lernplattform konzipiert und kann ohne Anpassung nicht direkt von o.g. Unternehmen eingesetzt werden. Möchte man das OLAT ohne Anpassung verwenden, so ergeben sich schnell mehrere Probleme.

OLAT bietet zunächst keine Möglichkeit, Währungen zu verwenden und Beträge auf Benutzerkonten zu verwalten, da diese Funktionen von einer Lernplattform nicht benötigt werden. Somit können auch keine Beträge überwiesen werden. Auch Funktionen zur Protokollierung bzw. revisionssicheren Speicherung von Transaktionen stehen somit nicht zur Verfügung.

Angebote und Gesuche könnten zwar theoretisch im OLAT erstellt werden, allerdings wird dies stets nur eine suboptimale Lösung darstellen. So könnte zweifelsohne das Forum oder Wiki eines Kurses (oder einer Gruppe) verwendet werden, um Angebote und Gesuche zu erstellen, es gibt jedoch keine fest definierte Form der Angebote (Titel, Beschreibung, Betrag, Gültigkeitszeitraum etc.). In Foren kann man zudem Beiträge i.A. nicht mehr löschen oder manipulieren, im Wiki kann jedes Mitglied Seiten ändern, also auch Angebote anderer Mitglieder. Die Suche nach Angeboten wäre nicht so einfach möglich, wie beispielsweise im Cyclos, wo gezielt nach bestimmten Angeboten gesucht werden kann. Es ist sicher leicht nachvollziehbar, dass mit den bestehenden Funktionalitäten das Erstellen und Suchen von Angeboten nicht, oder nur unzureichend, umgesetzt werden kann. Wikis, Foren und Chats könnten zwar die Kommunikation auf der Kooperationsplattform fördern, als virtueller Marktplatz bieten sie sich jedoch weniger an.

Da OLAT für einen anderen Zweck entwickelt wurde, würden beim Einsatz des Programms sehr viele Funktionen bereitstehen, die nicht benötigt werden. Für Mitglieder könnte das bedeuten, dass sie die Plattform als unübersichtlich empfinden bzw. mehr Zeit benötigen, um mit dem Programm vertraut zu werden.

4.4.2 Chancen

Trotz der im vorangegangen Abschnitt vorgestellten Probleme und Grenzen, ist es natürlich denkbar, OLAT durch entsprechende Erweiterung und Anpassung zu einer Software-Plattform zu modifizieren, die in regionalen Entwicklungsgesellschaften zum Einsatz kommen kann. Betrachtet man die Funktionalitäten von OLAT, wird man womöglich zunächst zu dem Schluss kommen, dass faktisch ein grundlegend

neues System entwickelt werden muss. Tatsächlich bietet OLAT allerdings schon sehr viele Komponenten und Funktionalitäten, die als Grundlage verwendet werden können. Dazu zählen:

- Nutzerprofil
- Rollen und Rechtesystem
- Kommunikationsmöglichkeiten (E-Mail, Chat etc.)
- Programmarchitektur bzw. Programmkern, insbesondere das Brasato-Framework

Vor allem der letzte Punkt scheint von hoher Bedeutung zu sein. In der Tat müssten zwar fast alle Funktionalitäten neu geschrieben werden, aber es existiert bereits ein sauber entwickelter Programmkern mit Datenbankanbindung, GUI-Framework, Sessionverwaltung, Request-Verarbeitung etc. Damit wäre die Anpassung von OLAT noch immer deutlich kostengünstiger als das Entwickeln einer komplett neuen Software.

4.4.3 Möglichkeit der Anpassung

Damit OLAT als Kooperationsplattform in einem Regionalentwicklungs-Netzwerk verwendet werden kann, müssen folgende Funktionen implementiert werden:

- Währungen
- Angebote und Gesuche
- Kontostand
- Transaktionen
- Verträge
- Die Rollen Kundenbetreuer und Kontakt

Hierbei wäre es zum Beispiel möglich, für Benutzer einen neuen statischen Tab einzurichten, indem auf der linken Seite dann die verschiedenen Funktionen wie Angebot suchen, Angebot erstellen, Kontostand anzeigen, Transaktion durchführen, Vertrag erstellen etc. aufgelistet sind. Für Anzeige-Funktionen (z.B. Suchresultate nach einem Suchvorgang, durchgeführte Transaktionen, abgeschlossene Verträge etc.) gibt es im OLAT bereits Tabellen-Komponenten (wie z.B. bei der Auflistung aller Gruppen), die dafür verwendet werden können. Für das Erstellen oder Suchen nach Angeboten, das Überweisen von Beträgen etc. würde sich ein klassisches Formular anbieten, welches im OLAT ebenfalls häufig verwendet wird. Für die Erweiterung müssen also letztendlich keine neuen grafischen Komponenten oder Strukturen erstellt werden.

Auf der Ebene der Geschäftslogik müssen dann die entsprechenden Funktionalitäten implementiert werden. Die Datenbank muss um entsprechende Tabellen erweitert werden. Hierbei würde sich z.B. eine Tabelle für Währungen, Angebote, Verträge, Kontostand sowie durchgeführte Transaktionen (eine Art Historie) anbieten. Auch für Branchen sollte eine eigene Tabelle existieren, wobei berücksichtigt werden muss,

dass Branchen auch Unterbranchen enthalten können. Im Anschluss müssen entsprechende Entitätsklassen erstellt werden und über das Hibernate persistent gemacht werden. Zusätzlich müssen dann die entsprechenden Manager und Controller programmiert werden, damit die Aktionen entsprechend verarbeitet bzw. durchgeführt werden können.

Damit zeigt sich, dass der Aufwand zwar relativ groß erscheint, für den Programmierer allerdings relativ wenig Schwierigkeiten und Probleme zu erwarten sind. Zudem kann er sich beim Erweitern der Software an den bereits implementierten Programmfunktionen von OLAT orientieren und auf die relativ gute Dokumentation zurückgreifen.

Lediglich das Implementieren der Rollen Kontakt und Kunde erscheint zunächst etwas schwierig. Im nächsten Abschnitt soll jedoch eine Lösung vorgestellt werden, die gerade dies realisiert hat.

4.4.4 Geschäftsvorgang des Kundenbetreuers

Am Lehrstuhl für Betriebliche Informationssysteme der Universität Leipzig hat Tom-Michael Hesse im Juni 2009 einen Prototyp vorgestellt, der den Geschäftsvorgang eines Kundenbetreuers der Anhalt-Dessau AG vollständig abdeckt und dabei das OLAT als Grundlage verwendet. Die Ideen und Konzepte der Arbeit, die am 15. Juni in einem Vortrag vorgestellt wurde, sollen in diesem Abschnitt kurz erläutert werden. Die Folien zu dem Vortrag stehen auf der Website http://bis.informatik.uni-leipzig.de/de/Forschung/EEE/Seminar zur Verfügung.

Aus der Entwurfsbeschreibung (Abbildung 6) lässt sich erkennen, dass der Record die zentrale Klasse des entwickelten Systems ist, die in enger Beziehung zu anderen grundlegenden Funktionsgruppen (z.B. Kontakt, Adresse, Branche, Person etc.) steht. Ein Record-Objekt soll dabei einen Kunden und einen Kundenbetreuer zusammenfassen, da es zu jedem Barterkunde einen Kundenbetreuer gibt. Von dem Barterkunden und dem Kundenbetreuer ist die Klasse Kontakt abgeleitet, von Kontakt sind wiederum die elementaren Klassen Adresse und Person abgeleitet. Es entsteht eine hierarchische Struktur, wie sie in Abbildung 7 zu sehen ist. Dabei steht der Record an oberster Stelle, der alle relevanten Daten des Kundenbetreuers und des Kunden akkumuliert.

Jede Funktionsgruppe (Adresse, Person, Kontakt etc.) besteht meist aus einem Interface für die entsprechende Entitätsklasse (z.B. Person), einer implementierenden Entitätsklasse (z.B. Person und PersonImpl) sowie einem Manager-Interface und einer implementierenden Managerklasse (z.B. PersonManager und PersonManagerImpl). Die Entitätsklassen enthalten die entsprechenden Daten und sind somit über das Hibernate an die Datenbank angeschlossen. Die Manager-Klassen verwalten alle Instanzen der Entitätsklassen und bieten entsprechende Funktionen zum Auflisten,

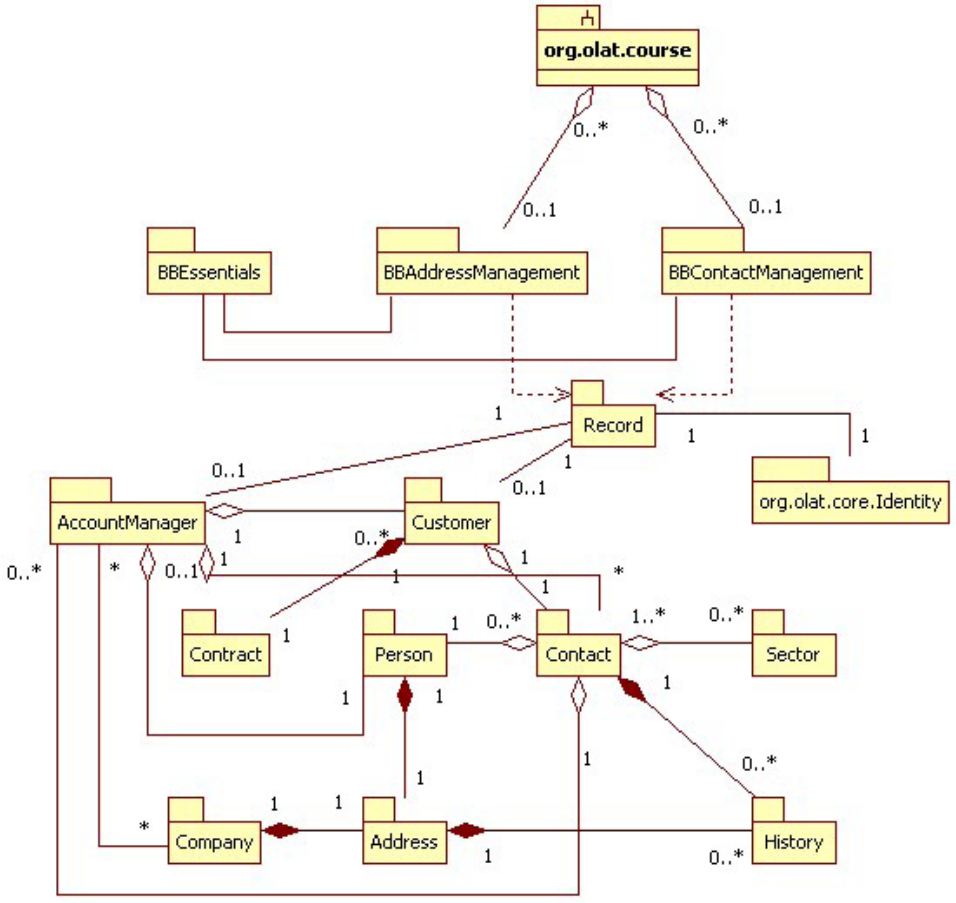

Abbildung 6: Übersicht über das gesamte Produkt, Januar 2009. Aus: Entwurfsbeschreibung der Kundenbetreuer-Erweiterung für die Kooperationsplattform, http://www.informatik.uni-leipzig.de/~graebe/Texte/Hesse-09.pdf

Suchen, Einfügen, Bearbeiten und Löschen entsprechender Instanzen.

Oberhalb des Record-Bausteins liegen BBAddressManagement und BBContactManagement, wobei das BB für Building Block steht. Diese beiden Bausteine enthalten die Controller und View-Klassen für das Verwalten von Adressen und das Verwalten von Kontakten. Sie orientieren sich dabei streng am Aufbau eines OLAT-Kurses [19].

In dem Vortrag vom 15. Juni wurde gezeigt, dass eine funktionierende Plattform zur Verwaltung von Kontakten seitens eines Kundenbetreuers der Anhalt Dessau AG entwickelt wurde. Das zeigt auch, dass das OLAT grundsätzlich zu einer Kooperationsplattform für Regionalentwicklungs-Netzwerke verwendet werden kann.

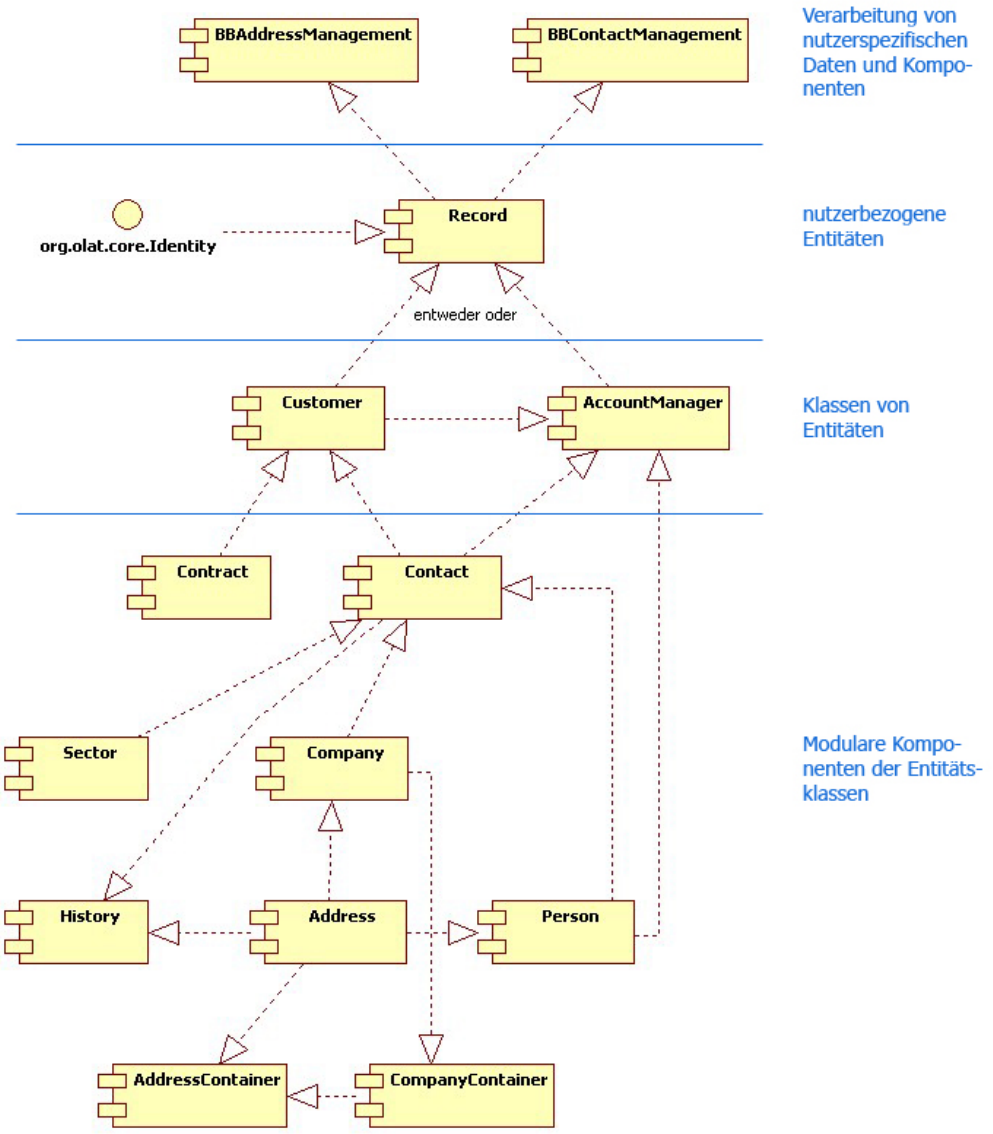

Abbildung 7: Schichtenarchitektur der Daten im Record, Januar 2009. Aus: Entwurfsbeschreibung der Kundenbetreuer-Erweiterung für die Kooperationsplattform, http://www.informatik.uni-leipzig.de/~graebe/Texte/Hesse-09.pdf

5 Vergleich der beiden Softwaresysteme

5.1 Vergleich

5.1.1 Einführung

OLAT und Cyclos erscheinen zunächst als zwei Webapplikationen, die unterschiedlicher nicht sein könnten. In der Tat hat die Analyse der beiden Systeme ergeben, dass es vor allem gravierende Unterschiede bezüglich der Zielgruppe, des Funktionsumfangs und des technischen Aufbaus gibt. In diesem Abschnitt sollen beide Systeme miteinander verglichen und damit Unterschiede und Gemeinsamkeiten aufgezeigt werden. Im zweiten Teil des Kapitels soll der Frage nachgegangen werden, welches System für regionale Entwicklungsnetzwerke besser geeignet scheint bzw. welches System als Grundlage verwendet werden sollte, um es dann an die speziellen Bedürfnisse des Unternehmens anzupassen.

5.1.2 Grundlagen

OLAT und Cyclos sind zwei kooperative Plattformen, die als Webanwendung erstellt wurden und somit über einen Browser ausgeführt werden. In beiden Systemen muss sich ein Nutzer registrieren, bevor er auf die einzelnen Funktionen zugreifen bzw. aktiv am System teilnehmen kann. Dabei gibt es in beiden Systemen mehrere Rollen. Im Cyclos sind es im Grunde nur drei Rollen (Member, Broker und Admin), im OLAT sind es insgesamt 6 Rollen, das heißt, es gibt eine feinere Abstufung bzw. Hierarchie. Insgesamt unterscheiden sich die beiden Plattformen aus dieser Sicht jedoch kaum und folgen damit den allgemeinen Richtlinien, die auch bei nahezu jeder anderen Web-Plattform angewendet werden.

OLAT wird bereits seit 1998 an der Universität Zürich entwickelt, ist also damit mehr als 10 Jahre alt. Die Plattform wird mittlerweile an vielen Institutionen eingesetzt. Cyclos ist deutlich jünger und scheint noch nicht die Bedeutung zu besitzen, die OLAT bereits erlangt hat. Aus dieser Sicht steht Cyclos also deutlich hinter dem OLAT. Andererseits zeigt die Liste der Unternehmen, die Cyclos bereits verwenden, dass die Bedeutung der Software relativ stark anwächst.[30] Für beide Systeme gilt dabei, dass sie Open Source sind und noch immer weiterentwickelt werden. Die neuste Version von Cyclos ist die Version 3.5, von OLAT ist es die Version 6.1.

[30] http://project.cyclos.org/index.php?option=com_content&task=view&id=233&Itemid=236

5.1.3 Zielgruppe

Cyclos und OLAT sind für unterschiedliche Zielgruppen konzipiert. Cyclos ist für regionale Entwicklungsgesellschaften konzipiert, deckt allerdings nicht sämtliche Anforderungen ab, die solche Netzwerke besitzen können. Zwar bietet Cyclos nahezu alle Funktionen, die für Regionalgeld-Gesellschaften erforderlich sind, und Regionalgeld und Tauschgeschäfte sind immerhin die momentan wesentlichsten Bestandteile eines regionalen Entwicklungsnetzwerks, es wurde aber am Beispiel der Anhalt Dessau AG gezeigt, dass es auch andere Funktionen und Ideen gibt, die in das Konzept solcher Unternehmen passen, und die bislang von Cyclos nicht unterstützt werden.

OLAT hat einen ganz anderen Nutzerkreis bzw. eine ganz andere Zielgruppe. OLAT wurde als reine Lernplattform entwickelt und soll insbesondere an Universitäten, Hochschulen und einfachen Schulen, weniger vielleicht auch an Institutionen und allgemeinen Einrichtungen, eingesetzt werden. Neben dem interaktiven Lernen in Gruppen und Kursen spielt auch die Verwaltung von Personen eine gewisse Rolle. Damit kann OLAT also auch in Unternehmen eingesetzt werden, wo es nicht mehr als Lernplattform dient. Die Bedeutung von OLAT als reines Verwaltungsprogramm ist jedoch eher gering.

Cyclos ist also für regionale Entwicklungsnetzwerke konzipiert, OLAT hingegen für Lern-Netzwerke. Beide Plattformen haben aber gemeinsam, dass sie i.A. nur in einer bestimmten Region eingesetzt werden (Landkreis, Bundesland etc.). Dabei kann die Region von Cyclos deutlich größer sein (z.B. mehrere Landkreise oder Bundesländer) als der von OLAT, wo im Minimalfall nur eine Institution betroffen ist. In Einzelfällen wäre aber auch für das OLAT ein sehr weiter (z.B. landesweiter) Einsatz möglich.

5.1.4 Bedienbarkeit und Darstellung

Bedienbarkeit und Darstellung sind bei beiden Plattformen relativ einfach gehalten. Trotzdem erscheint das Cyclos in der Bedienbarkeit etwas komfortabler. Hier besitzt der Benutzer lediglich ein Menüsystem, das einige Menüpunkte mit Unterpunkten enthält. Beim OLAT gibt es statische Tabs mit einer Reihe von Untermenüs auf der linken Seite und teilweise Tools auf der rechten Seite sowie allgemeine Funktionen im Kopfbereich. Zudem werden manche Inhalte in neuen Tabs (dynamische Tabs) oder im Content Bereich angezeigt. Obwohl diese Darstellung durchaus nützlich ist bzw. bei dem großen Funktionsumfang auch durchaus notwendig erscheint, ist OLAT nicht so übersichtlich und einfach zu bedienen wie Cyclos. Die zahlreichen Funktionalitäten und Konzepte (Lernressourcen, Kurse, Lerngruppen, Arbeitsgruppen, Rechtegruppen etc.) tragen ebenso dazu bei, dass ein neues Mitglieder vermutlich mehr Zeit benötigt, sich auf der Plattform zurecht zu finden bzw. mit ihr arbeiten zu können.

Die Darstellung ist beim Cyclos etwas schlichter als beim OLAT, sonst unterschei-

den sich beide Plattform diesbezüglich jedoch kaum. Einige Seiten im OLAT können ziemlich viel Inhalt enthalten, z.B. der Home-Bereich. Aus dieser Sicht kann das Cyclos an manchen Stellen etwas übersichtlicher als das OLAT sein.

Cyclos scheint also zumindest in der Bedienbarkeit etwas mehr Vorteile zu bieten als OLAT. Es muss dabei aber auch bedacht werden, dass bei einer Erweiterung von OLAT nicht viel an dem Gesamtlayout und der Bedienbarkeit geändert werden kann (bzw. nur mit großem Aufwand) und somit für Mitglieder der Plattform stets mit einem höheren Einarbeitungsaufwand gerechnet werden muss, als bei Cyclos.

5.1.5 Funktionalitäten

Entsprechend der unterschiedlichen Zielgruppen unterscheiden sich beide Plattformen deutlich in Hinsicht der Funktionalitäten.

Cyclos ist direkt für regionale Entwicklungsgesellschaften konzipiert und bietet damit weitgehend alle grundlegenden Funktionen, die hierfür notwendig sind, allen voran die Möglichkeit, Angebote und Gesuche zu erstellen und zu suchen, Transaktionen durchzuführen und den Kontostand zu verwalten. Auch das Verwenden einer Ersatzwährung, die Grundlage solcher Tauschgeschäft ist, ist bereits implementiert.

OLAT ist für solche Netzwerke hingegen nicht ausgelegt und muss zunächst um einige Funktionen erweitert werden, um in regionalen Entwicklungsgesellschaften zum Einsatz kommen zu können. Immerhin bietet es bereits ein Nutzerprofil und weitreichende kommunikative Möglichkeiten, z.B. Chat, Wiki, Forum, E-Mail etc. Ein interner Nachrichtendienst, wie er im Cyclos realisiert wurde, fehlt zwar, es gibt jedoch im OLAT grundsätzlich die Möglichkeit, E-Mails an andere Mitglieder zu versenden, was diesem Nachrichtendienst sehr nahe kommt.

Trotz dass Cyclos direkt für regionale Entwicklungsgesellschaften zugeschnitten ist, hatte sich gezeigt, dass einige durchaus relevanten Funktionen bislang nicht implementiert sind. Dazu zählt z.B. das Verwenden mehrere Währungen und das Abschließen von Verträgen vor dem eigentlichen Tauschgeschäft. In Bezug zur Anhalt Dessau AG fehlt das Verwalten von Kontakten seitens eines Kundenbetreuers.

Hiermit zeigt sich, dass unter Umständen beide Systeme erweitert werden müssen, bevor sie in einem Regionalentwicklungsunternehmen verwendet werden können. Der Erweiterungsaufwand ist bei Cyclos allerdings geringer als beim OLAT.

5.1.6 Programmarchitektur

Sowohl OLAT als auch Cyclos weisen eine Schichtenarchitektur auf, die der im 2. Kapitel vorgestellten 3-Schichten-Architektur entspricht und bei einem typischen Workflow zunächst von oben nach unten und anschließend von unten nach oben durchlaufen wird. Die Datenhaltung geschieht durch Verwendung eines Datenbanksystems (z.B. MySql), das über Hibernate an das Programm angeschlossen ist.

Unterschiede zwischen beiden System gibt es in den einzelnen Schichten bzw. in der Umsetzung grundlegender Funktionalitäten. So wird im Cyclos das Struts-Framework verwendet, um nach einem Browser-Request den entsprechenden Controller aufzurufen, während im OLAT Dispatcher-Klassen den Aufruf des korrekten Controllers gewährleisten. Die Ausgabe geschieht im Cyclos mittels Tiles und den JSP-Pages, während sie im OLAT über ein selbst von den Programmierern erstelltes Framework in HTML gebracht wird. Im Cyclos können Service-Klassen (Model) nur über ein DAO-Objekt auf die Datenbank zugreifen, im OLAT können die Manager-Klassen hingegen direkt auf die Datenbank zugreifen.

Auch wenn es also einige nennenswerte Unterschiede im Programmaufbau bzw. in der Funktionsweise gibt, erscheinen beide Systeme relativ gut durchdacht und nachvollziehbar umgesetzt. Auch realisieren beide Plattformen das MVC-Konzept.

5.1.7 Quellcode

Beide Systeme sind in Java geschrieben und weißen eine relativ flache, gut strukturierte Klassenhierarchie auf. OLAT ist deutlich umfangreicher als Cyclos und speichert auch zahlreiche property-, html-, css- und xml-Dateien in den einzelnen Packages. Cyclos speichert i.A. nur die einzelnen Java-Klassen und ein paar wenige xml-Dateien in den Packages; damit erscheint die Klassenhierarchie übersichtlicher und weniger komplex.

Der OLAT- und Cyclos-Quellcode folgen ungefähr den allgemeinen Java-Konventionen, allerdings gibt es deutliche Unterschiede in der Dokumentation. OLAT ist deutlich besser dokumentiert als Cyclos. Zwar ist in beiden Softwaresystemen die JavaDoc-Dokumentation unvollständig, in OLAT sind aber zumindest die wichtigsten Klassen vollständig dokumentiert. Zudem gibt es mehr Inline-Kommentare und vor allem umfangreichere Programmspezifikationen.

5.1.8 Erweiterbarkeit

Bei der Frage, welches Softwaresystem einfacher zu erweitern ist, scheint OLAT auf Grund der relativ guten Dokumentation und den zahlreichen Programmspezifika-

tionen die günstigere Wahl zu sein. Allerdings muss dabei berücksichtigt werden, dass OLAT deutlich komplexer als Cyclos ist. Eine zeitintensive Einarbeitung in die Software ist daher sowohl beim OLAT als auch beim Cyclos zu erwarten. OLAT bietet jedoch zumindest den Vorteil, dass mit der Version 6.1 ein längst ausgereiftes und vielfach getestetes und verbessertes System vorliegt. Insofern ist anzunehmen, dass OLAT insgesamt sicherer ausgeführt werden kann und eine Erweiterung weniger Komplikationen verursachen wird als bei Cyclos. Die Erweiterung wird zudem durch die zahlreichen integrierten Tools zum Debuggen vereinfacht, die bei Cyclos weitgehend fehlen. Ein weiterer Vorteil von OLAT ist das Erstellen der Ausgabe, die durch das Brasato-Framework relativ einfach ist. Dies ist ein großer Unterschied zum Cyclos, wo für jede neue Seite eine neue JSP-Page mit entsprechender JS-Datei und entsprechender Eintragung in die struts-config-Datei notwendig ist. Das impliziert auch, dass ein Programmierer sich ggf. zunächst in das Konzept der JSP-Pages einarbeiten muss, bevor er überhaupt die Erweiterung am Cyclos vornehmen kann.

Damit scheint OLAT in der Tat die besseren Voraussetzungen für eine Programmerweiterung zu bieten.

5.2 Die Systeme im Einsatz von Regionalentwicklungsunternehmen

In diesem Abschnitt soll schließlich die Frage erörtert werden, welches System für den Einsatz in den im 1. Kapitel aufgezeigten regionalen Entwicklungsgesellschaften besser geeignet ist. Legt man den Fokus dabei auf die Anhalt Dessau AG, so hat sich bereits gezeigt, dass sowohl Cyclos als auch OLAT für die Funktionalitäten nicht ausreichen. Beide Softwaresysteme müssen zunächst erweitert werden.

Da die Anhalt Dessau AG jedoch genau in die Zielgruppe von Cyclos Software Projekts fällt, sind die meisten grundlegenden Funktionalitäten bereits im Cyclos implementiert. Es müssen nur verhältnismäßig wenig neue Funktionalitäten implementiert werden. Ganz anders sieht es dabei bei OLAT aus. Hier müssen, wie bereits aufgezeigt wurde, fast alle Funktionalitäten neu implementiert werden, da OLAT als Lernplattform zunächst kaum für regionale Entwicklungsgesellschaften geeignet ist. Die folgende Übersicht vermittelt einen Überblick, was im Cyclos hinzugefügt werden müsste, damit es von der Anhalt Dessau AG eingesetzt werden könnte.

Zu ergänzende Funktionen im Cyclos:

- Abschließen und revisionssicheres Speichern von Verträgen
- Hinzufügen zweier Rollen: Kontake und Kundenbetreuer
- Mehrere Währungen unterstützten (Vergleich über Wechselkurs)

Im OLAT hingegen, das wurde im vorangegangenen Kapitel gezeigt, müssen fast alle grundlegenden Funktionalitäten wie Währung, Angebot, Vertrag, Transaktion,

Kontostand etc. ergänzt werden. Einige Elemente sind im OLAT jedoch bereits implementiert und müssen daher nicht hinzugefügt werden: Profil, einige Kommunikationsmittel (hier wäre die Verwendung eines internen Nachrichtendienstes überlegenswert, aber nicht zwangsweise notwendig) sowie die gesamte grundlegende Programmarchitektur bzw. der Programmkern (GUI, Datenbankanbindung, Dateisystem etc.).

Somit scheint es fast offensichtlich, dass die Erweiterung von Cyclos deutlich weniger Zeit und Aufwand in Anspruch nehmen wird als die Erweiterung am OLAT. Hierbei kommt begünstigend hinzu, dass einige Erweiterungspunkte im Cyclos auch relativ einfach zu realisieren sind. So können Kontake als *pending members* angesehen werden und Kundenbetreuer als *broker*; entsprechend gering scheint somit der Änderungsaufwand. Auch das Verwenden mehrerer Währungen scheint nicht besonders schwierig zu sein, bzw. wird in einer neuen Version von Cyclos evtl. bereits implementiert sein. Insgesamt scheint somit die Erweiterung von Cyclos bei weitem nicht so aufwendig wie bei OLAT.

Von der Bedienung und dem Layout her scheint Cyclos ebenfalls besser geeignet zu sein. Es ist auf Grund der weniger komplexen Funktionen relativ übersichtlich aufgebaut, während OLAT auf den ersten Blick etwas unübersichtlich erscheint. Natürlich kann man OLAT so anpassen, dass alle Funktionen entfernt sind, die nicht zu der Kooperationsplattform gehören (das wäre ohnehin empfehlenswert), was jedoch zusätzlichen Aufwand bedeutet.

Natürlich bietet aber auch das OLAT einige wesentliche Vorteile. Wie schon erwähnt, ist die Erweiterung bisweilen einfacher als bei Cyclos, es gibt bessere Unterstützung durch Debugging-Tools und Programmdokumentation, die Software ist ausgereifter und ermöglicht das Verwenden einiger Funktionen, die Cyclos nicht bietet (z.B. Forum, Wiki, Chat, Terminkalender). Insgesamt scheint jedoch Cyclos die bessere Wahl zu sein. Zwar hängt es letztendlich stark von den Präferenzen des Kunden ab, welches System letztendlich zum Einsatz kommen soll, da am Cyclos der Erweiterungsaufwand jedoch relativ gering ist, scheint Cyclos gegenüber dem OLAT stärkere Vorteile aufzuweisen. Letztendlich würden somit auch die Kosten geringer sein, als bei einer Erweiterung von OLAT.

OLAT und Cyclos weisen, das hat sich gezeigt, unterschiedliche Vor- und Nachteile auf, können aber gleichermaßen als Basis für eine Kooperationsplattform in einem Regionalentwicklungs-Netzwerk eingesetzt werden. Eins sollte jedoch deutlich geworden sein: Dass das Cyclos Projekt bereits ein leistungsstarkes Softwareprodukt ist, welches ein hohes Maß an IT-Unterstützung für ein regionales Entwicklungsnetzwerk bietet.

6 Zusammenfassung

In der vorliegenden Arbeit wurden zwei mächtige Web-Plattformen untersucht, die beidermaßen in Regionalentwicklungs-Netzwerken eingesetzt werden können. Ausgehend von der Frage, was Regionalentwicklungs-Netzwerke sind, welche typischen Geschäftsvorfälle damit verbunden sind und welche Anforderungen an ein Softwareprodukt bestehen, welches in diesen Netzwerken zum Einsatz kommen soll, wurde zunächst der allgemeine Aufbau von Web-Plattformen dargelegt, bevor dann die beiden Programme Cyclos und OLAT näher untersucht wurden.

Bei der Untersuchung wurde bei beiden Softwareprodukten gleichermaßen auf den Funktionsumfang einerseits und den technischen Aufbau sowie die Erweiterungsmöglichkeiten andererseits eingegangen. Nur durch die Betrachtung beider Schwerpunkte war es somit möglich, ein Urteil zu bilden, welche Plattform letztlich eher geeignet ist, um in regionalen Entwicklungsnetzwerken zum Einsatz zu kommen, insbesondere an der Anhalt Dessau AG, die in dieser Arbeit einen besonderen Stellenwert hatte.

Beim Vergleich der beiden Programme hat sich gezeigt, dass beide Systeme unterschiedliche Vor- und Nachteile aufweisen. Beide Systeme besitzen auch deutliche Differenzen bezüglich des Funktionsumfangs, des technischen Aufbaus und der Zielgruppe. OLAT ist eine reine Lernplattform und damit zunächst nicht für regionale Entwicklungsnetzwerke konzipiert. Es wurde jedoch gezeigt, dass OLAT ein leistungsfähiges, bewährtes Softwareprodukt ist, dessen Erweiterung dank guter Dokumentation und zahlreichen integrierten Tools (Debugging-Tools) sowie vorgefertigten Komponenten (Brasato-Framework, Rollensystem etc.) relativ einfach erscheint. Eine Erweiterung des noch in der Entwicklung befindlichen Cyclos ist hingegen deutlich aufwendiger. Die Software ist noch nicht so ausgereift wie OLAT, weist Mängel in der Dokumentation auf und bietet weniger Unterstützung für Programmierer. Andererseits besitzt Cyclos fast alle wichtigen Funktionalitäten, die zu den Anforderungen von regionalen Entwicklungsgesellschaften gehören. Die Erweiterung von Cyclos erscheint somit zwar schwieriger als die von OLAT, der Gesamtaufwand ist jedoch vermutlich geringer, da ein Großteil an Funktionalitäten bereits zur Verfügung steht.

Das Ergebnis der Arbeit ist somit, dass eine Anpassung von Cyclos für regionale Entwicklungsunternehmen bzw. die Anhalt Dessau AG eher geeignet ist, als eine Anpassung von OLAT. Nichtsdestoweniger wurde gezeigt, dass grundsätzlich beide Plattformen in Regionalentwicklungsnetzwerken eingesetzt werden können und dabei unterschiedliche Vor- und Nachteile aufweisen. Letztlich kommt es somit auch auf die individuellen Vorstellungen und Anforderungen der jeweiligen Unternehmen an, welche Software für die IT-Unterstützung die günstigere Wahl ist.

Literatur

[1] Cyclos - Open Source complementary currency software. http://project.cyclos.org/

[2] OLAT - The Open Source LMS. http://www.olat.org/

[3] Initiative Dessau - Arbeit für Anhalt e.V. Eine Initiative für mehr Beschäftigung. http://www.ini-dessau.de/

[4] RegioStar e.G. - „...weil uns die Region am Herzen liegt“. http://www.regiostar.com/

[5] Regiogeld - Der neue Herzschlag der Region. http://www.regiogeld.de/urstromtaler.html

[6] Wikipedia, The Free Encyclopedia, Artikel *Local Exchange Trading Systems*. http://en.wikipedia.org/wiki/Local_Exchange_Trading_Systems

[7] North London LETS - How It Works. http://www.nllets.org.uk/explain.htm

[8] Talente-Tausch Graz. http://talentetauschgraz.at/

[9] Talente-Tausch Graz, Regeln. Fassung Jänner 2009. http://talentetauschgraz.at/regeln.pdf.

[10] Volker Turau, Krister Saleck, Christopher Lenz (2004): Web-basierte Anwendungen entwickeln mit JSP 2 (2. vollständig überarbeitete Auflage), dpunkt.verlag

[11] Hibernate - Relational Persistence for Java and .NET. https://www.hibernate.org/

[12] Spring Source - About Spring. http://www.springsource.org/about

[13] The Apache Software Foundation - Struts. http://struts.apache.org/

[14] Code Conventions for the Java^{TM}Programming Language. http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html

[15] OLAT 6 - Benutzerhandbuch. http://www.olat.org/website/en/download/help/OLAT_6_1_Manual_DE_online_090306.pdf

[16] Projekthandbuch „Funktionelle und softwaretechnische Analyse sowie Dokumentation der Lernplattform OLAT Version 3.1.4“, Technische Universität Chemnitz. http://www.olat.org/downloads/presentations/OLAT_Projekthandbuch_mitAnhang.pdf

[17] OLAT 6 - Funktionsübersicht. http://www.olat.org/website/en/download/OLAT_6_0_Funktionsuebersicht.pdf

[18] OLAT Systemdokumentation - Die System- und Softwarearchitektur des LMS OLAT im Überblick. http://www.frentix.com/de/olat/docu/OLAT-Systemdokumentation.pdf

[19] Entwurfsbeschreibung der Kundenbetreuer-Erweiterung für die Kooperationsplattform. http://www.informatik.uni-leipzig.de/~graebe/Texte/Hesse-09.pdf