Diplomarbeit

Bülent Ulas

Entwicklung einer Software für Online-Befragungen mit interaktiven evolutionären Algorithmen

Fakultät Technik und Informatik
Studiendepartment Informatik

Hamburg, 12. März 2008

Bülent Ulas

buelent@ulas.de

Entwicklung einer Software für Online-Befragungen mit interaktiven evolutionären Algorithmen

Diplomarbeit eingereicht im Rahmen der Diplomprüfung
im Studiengang Softwaretechnik
am Studiendepartment Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Wolfgang Gerken
Zweitgutachter: Prof. Dr. Thorsten Teichert

Abgegeben am 12. März 2008

Bülent Ulas

Thema der Diplomarbeit

Entwicklung einer Software für Online-Befragungen mit interaktiven evolutionären Algorithmen

Stichworte

Evolutionäre Algorithmen, Evolutionäre Conjoint, Interaktive Algorithmen, Marktforschung

Kurzzusammenfassung

Die Integration von Konsumenten in den Neuentwicklungsprozess ist unabdingbar für ihren Markterfolg.
Die in der Praxis gängigen Marktforschungsmethoden basieren auf Konsumentenbefragungen, um ihre angegebenen Präferenzen zu analysieren.
Interaktive evolutionäre Algorithmen sind ein neuartiges Befragungsinstrument, welches insbesondere für online Befragungen geeignet ist.
Ihr Einsatz in Marketinganwendungen hat einen hohen Beitrag sowohl aus wissenschaftlicher als auch als praxeologischer Sicht.

Bülent Ulas

Title of the paper

Development of an online survey application based on interactive evolutionary algorithms

Keywords

online survey application, interactive evolutionary algorithms, evolutionary conjoint, conjoint, market research

Abstract

The integration of consumers in new development process is inalienable for the market sucess. in the practice popular market research based on consumer researches to analyse the named preferences. Interactive evolutionary algorithm are an innovative survey instrument witch is particulary appropriate for online surveys. The action in marketing use has a high contribution as well in science as in praxeological point of view

Inhaltsverzeichnis

1 Einleitung

Aufgrund gesättigter Märkte stehen Unternehmen zunehmend unter einem steigenden Kosten- und Erfolgsdruck. Eins der Ziele der Unternehmen ist es, ihre bestehenden Kunden möglichst langfristig an ihr Unternehmen zu binden, sowie Neukunden zu akquirieren. Um dieses Ziel erreichen zu können, bedarf es einer fundierten Marketinganalyse, sowie einem Optimierungsverfahren, das durch sein System die gesetzte Zielfunktion minimiert oder maximiert.
Insbesondere der hohe Datenbedarf und die schwierige Datenbeschaffung stellen für viele Unternehmen eine Barriere dar. In Bezug auf das Optimierungsverfahren werden immer mehr Computertechnologien eingesetzt. Ein vordergründiges Interesse der Unternehmen ist die Verbesserung von Kosten- oder Rohstoffoptimierung.
Die vorliegende Arbeit wird im Wesentlichen einen Ansatz untersuchen, bei dem die Optimierung von Produktkonfigurationen aus Konsumentensicht im Vordergrund steht.
Die konkrete Vorgehensweise sowie die Beweggründe für diesen Ansatz werden in den nachfolgenden Kapiteln diskutiert. Geschrieben wird die Arbeit in Zusammenarbeit mit dem Arbeitsbereich Marketing und Innovation der Universität Hamburg (AMI).

1.1 Problemstellung

Unternehmen die einen hohen Marktanteil für sich gewinnen sowie eine hohe Kundenbindung erreichen möchten, benötigen eine gezielte Marketingstrategie. Die Einführung von innovativen Produkten stellt einen Ausgangspunkt zur Erreichung eines hohen Marktanteils dar. Jedoch erfolgt vor der Einführung eines neuen Produkts ein Strategisches Marketing, dass Ziel, Konzept und Durchführung verfolgt. Zudem müssen produktpolitische Maßnahmen getroffen werden. Ein Produkt sollte den Bedürfnissen der Abnehmer entsprechen und als Leistung eines Anbieters gesehen werden. Jedes Produkt durchläuft einen Lebenszyklus, der aus fünf Phasen besteht: Einführung, Wachstum, Reife, Sättigung, Degeneration. [Mef00, Seite 339] Dieser Lebenszyklus kann aber durch Flops unerwartet beendet werden. Das Floprisiko für ein neues Produkt kann in Extremfällen bis zu 70% erreichen[Mef00, Seite 378]. Gründe für so genannte Flops, können folgende sein: Falsche Wahl des Einführungsproduktes, unzureichende Distributionspolitik, zu früh oder zu späte Einführungswerbung, unausgereiftes Produkt, unzureichende Planung (Netzplantechnik), falsche Produktpreise und unzureichender Kundendienst.
Um Flops zu vermeiden wird Marktforschung betrieben. Sie wird als systematisches und methodisch einwandfreie, laufende oder fallweise Untersuchung eines Marktes verstanden und hat das Ziel, Entscheidungen in diesem Bereich zu treffen und zu erklären.
Eine in der Marktforschung eingesetzte Methode zum Ermitteln der Präferenzen der Konsumenten ist die Conjoint-Analyse. Unter dem Begriff der Conjoint-Analyse wird eine Reihe von multivariaten Untersuchungsansätzen zusammengefasst. Bei der Conjoint Analyse werden nicht

Einzelurteile über spezifische Eigenschaften eines Produktes zu einer Gesamtbeurteilung zusammengesetzt, sondern insbesondere Gesamturteile erhoben, aus denen der Beitrag einzelner Eigenschaften zu diesem Urteil errechnet wird [BEPW05, Seite 558]. Bei dieser Analyse werden Kombinationen aus den Eigenschaften und deren Ausprägungen des Produktes gebildet, was ein großer Nachteil für diese Analyse darstellt. Bei einem Produkt, für das fünf Eigenschaften und drei Ausprägungen ausgewählt wurde, sind die Kombinationen $3^5 = 243$ und bei vier Eigenschaften 1024. Dies hat zur Folge, dass nur wenige Eigenschaften und Ausprägungen in die Analyse einbezogen werden. Darüber hinaus stehen die Kombinationen vor der Durchführung der Befragung fest und können nicht während der Befragung geändert werden.
Deshalb besteht der Anspruch, eine Anwendung zu entwickeln, die diese Problematik durch geeignete Algorithmen wie z. B. evolutionäre Algorithmen und Hilfsmethoden wie z. B. statistische Auswahlmethoden zumindest teilweise umgeht.

1.2 Zielsetzung

Im Rahmen dieser Arbeit soll einen Entwurf für eine Software zur Durchführung von Online-Befragungen gemacht werden. Der entstandene Entwurf muss anschließend implementiert werden, dabei muss die Anwendung für die Vorbereitung der Befragungen die Evolutionären Algorithmen einsetzen. Nachfolgend seien die Anforderungen an die zu entwickelnde Online-Befragung-Anwendung aufgezählt.

- Befragungen müssen online durchgeführt werden können.
- Sie soll für Befragungen, bei denen das Aussehen eines Produktes optimiert werden soll, universell einsetzbar sein.
- Die Erzeugung, die Bearbeitung der Befragungen und das Ansehen der Ergebnisse sollen gewährleistet werden.
- Die Möglichkeit, die Befragungen vor öffentlichem Zugriff zu schützen, soll angeboten werden.
- Sie soll über ein Web-Interface administrierbar sein.
- Die Befragungsdaten sollen persistent gespeichert werden.

Die Details der Anforderungen werden in Abschnitt 3.3 besprochen.

1.3 Aufbau der Arbeit

Die Arbeit ist in fünf Kapiteln unterteilt. Nach diesem Einleitungsteil wird in Kapitel 2 (Grundlagen) zunächst auf die Conjoint-Analyse im Allgemeinen eingegangen. Dem folgt eine Übersicht über die Evolutionären Algorithmen. Die Zusammenhänge zwischen der Conjoint-Analyse und den Evolutionären Algorithmen werden im Unterkapitel 2.3 erläutert. Abgeschlossen wird das zweite Kapitel mit einer Beschreibung weiterer Technologien, die im Rahmen dieser Arbeit von Bedeutung sein werden.

Im ersten Teil des dritten Kapitels wird die vorhandene Lösung Vorgestellt und eine Schwachstellenanalyse vorgenommen. Anschließend werden die Details der Anforderungen näher dargestellt. Das dritte Kapitel wird mit der Soll-Analyse abgeschlossen.
Kapitel 4 thematisiert den Entwurf und die Implementation der Online-Befragung-Anwendung. In Kapitel 4 wird zunächst der Entwurf der Online-Befragung-Anwendung vorgestellt. Dem folgt die Erklärung der Architektur und der Anwendungskomponenten. Die Realisierung der Online-Befragung-Anwendung wird im zweiten Teil des vierten Kapitel erläutert.
Kapitel 5 fasst die wichtigsten Erkenntnisse dieser Arbeit zusammen. Insbesondere werden die Anforderungen auf Erfüllung überprüft. Zum Schluss wird in einem Ausblick auf Erweiterungsmöglichkeiten der Online-Befragung-Anwendung eingegangen.

Im weiteren Verlauf dieser Arbeit wird die zu entwickelnde Anwendung “EvoMarketOnline” genannt.

2 Grundlagen

Die Online-Anwendung EvoMarketOnline soll die Nachteile der Conjoint-Analyse vermindern, in dem sie die Produkte mit Hilfe der Evolutionären Algorithmen verändert und darstellt.
In diesem Kapitel wird eine Reihe von Grundlagenthemen diskutiert. Eines dieser Themen ist die "Conjoint-Analyse", die in Abschnitt 2.1 besprochen wird. Anschließend folgt eine Erörterung der Themen "Evolutionäre Algorithmen", "Evolutionäre Conjoint" in den Abschnitten 2.2 und 2.3.
Dieses Kapitel wird mit der Erläuterung der "Technische Grundlagen" in Abschnitt 2.4 abgeschlossen.

2.1 Conjoint-Analyse

Mit dem Terminus Conjoint-Analyse oder auch Conjoint Measurement bezeichnet man eine Gruppe von multivariaten Untersuchungsansätzen, die ihren Ursprung aus dem Bereich der mathematischen Psychologie und der Psychometrie haben[Mef00]. In den 70er Jahren wurde die Conjoint-Analyse in der Marktforschung eingeführt. Seither erfreut sie sich insbesondere im Produktmarketing traditionell großer Beliebtheit.
Dieses Marktforschungsinstrument ist ein empirisches Verfahren, welches die Präferenzen von Testpersonen ermittelt. Bei der Conjoint Analyse werden Kombinationen aus den relevanten Eigenschaften eines Produktes gebildet. Die Befragten beurteilen nicht mehr die einzelne Eigenschaften, sondern deren Kombination bzw. das gesamte Produkt.
Mit Hilfe von geeigneten statistischen Verfahren können die Einflüsse der einzelnen Eigenschaften zurückgerechnet werden[BEPW05, Seite 558].
Darüber hinaus kann mit den Ergebnissen der Conjoint-Analyse die Clusteranalyse durchgeführt werden, deren Ziel die Konsumenten, die dieselbe Präferenzen haben, zu Gruppen zusammenfassen ist. Somit kann wertvolle Hinweise auf die Zielgruppe des untersuchten Produktes gewonnen werden.
Die Conjoint-Analyse wird bei der Neuproduktentwicklung häufig eingesetzt, um die Frage zu beantworten, wie ein neues Produkt in Hinsicht auf die Bedürfnisse des Marktes optimal zu gestalten ist. Durch diese Methode kann eine optimale Leistungsgestaltung sowie eine effiziente Platzierung am Markt erzielt werden.
Im Folgenden wird die Durchführung der Conjoint-Analyse anhand eines Vorgehensshema erläutert, dabei wird folgendes Beispiel zur Verdeutlichung zur Hilfe genommen.
Ein Spirituosenhersteller möchte ein neues Getränk auf dem Markt bringen. Um die Flasche des Getränkes nach den Wünschen zu gestalten, müssen folgende Eigenschaften und deren Ausprägungen in die Untersuchung einbezogen werden.

“Flaschenfarbe”	blau, grün, braun
“Etikettenfarbe”	gelb, schwarz, rot
“Etikettentext”	oben, mittig, unten

Durch diese Festlegung, drei Eigenschaften mit jeweils drei Ausprägungen, müssen insgesamt $3^3 = 27$ Kombinationen gebildet werden. Die Vorgehensweise der Conjoint-Analyse kann in fünf Schritten unterteilt werden (Vgl. [BEPW05, Seite 562] und [Mef00, Seite 402]):

- Festlegung der Eigenschaften und deren Prägungen.
- Erhebungsdesign.
- Bewertung der Stimuli.
- Schätzung der Nutzenwerte.
- Aggregation der Nutzenwerte

Festlegung der Eigenschaften und deren Prägungen

Dieser Schritt erfordert eine Reihe von Anforderungen [BEPW05, Seite 562]. In eine Conjoint-Analyse sollen nur die Eigenschaften, die für die Gesamtnutzenbewertung der Befragten von Bedeutung sind und für die Kunden mögliche Kaufentscsheidungskriterien bilden können, einfließen. Bei innovativen Produktideen ist es aber schwer festzulegen, welche Eigenschaften bzw. deren Ausprägungen für die Kaufentscheidung der Kunden relevant sind, z. B. sind aus dem obigen Beispiel die Etikettenfarbe oder Position des Etikettentextes der Flasche Kaufentscsheidungskriterien?
Der Hersteller muss die Möglichkeit haben, die Eigenschaften zu beeinflussen und dies zu realisieren.
Die Eigenschaften müssen unabhängig sein, so dass der Nutzen einer Eigenschaftsausprägung nicht durch die Ausprägungen anderer Eigenschaften beeinflusst wird.
Darüber hinaus soll die Annahme, dass eine Gesamtbeurteilung eines Produktes durch Summation aller Einzelurteile der als gegenseitig substituierbar angesehenen Eigenschaftsausprägungen ergibt, ermöglicht werden.

Erhebungsdesign

Im Rahmen der Festlegung des Erhebungsdesigns müssen folgende Entscheidungen getroffen werden:

1. Wie müssen Stimuli definiert werden: Profilmethode oder Zwei-Faktor-Methode?
2. Die Zahl der Stimuli muss festgelegt werden: Vollständiges Design oder reduziertes Design?

Vollständiges Design: Alle möglichen Stimuli werden zum Bewerten ausgewählt. Beim obigen Beispiel sind 27 Stimuli auszuwählen. Das vollständige Design ist nicht immer möglich, weil die Anzahl der Stimuli überproportional zu Eigenschaften und deren Ausprägungen steigt. Wenn z. B.

im obigen Beispiel noch eine Eigenschaft mit drei Ausprägungen in die Untersuchung einbezogen wird, sind die Anzahl der Stimuli $3^4 = 81$.
Zwei-Faktor-Methode: Bei dieser Methode werden die Ausprägungen von jeweils zwei Eigenschaften miteinander kombiniert.
Profil-Methode: Bei dieser Methode werden Stimuli als vollständige Objektprofile mit jeweils einer Ausprägung aller Merkmale gebildet.
Reduziertes Design: Eine angemessene Stimuli aus der möglichen Stimuli-Menge werden zum Bewerten ausgewählt.
Die Details dieser Schritt sind der Literatur zu entnehmen[BEPW05, Seite 564-570].

Bewertung der Stimuli

Die Conjoint-Analyse erfordert, dass die Befragten ihre Präferenzen offen legen, indem sie das Gesamt-Produkt bewerten. Die Offenlegung der Präferenzen durch die Befragten kann über verschiedene Wege erfolgen.
Rangreihung: Bei diesem Verfahren werden die Befragten aufgefordert, die Stimuli, die ihre individuelle Präferenz widerspiegelt, in eine Reihenfolge zu bringen. Bei einer hohen Anzahl der Stimuli kann das schnell dazu führen, dass der Befragte überfordert wird.
Profilpaarmethode: Die Stimuli werden den Befragten paarweise zur Beurteilung vorgelegt. Diese Methode hat den Vorteil, dass der Befragte nicht mehr überfordert wird. In der Regel können aber nicht alle möglichen Kombinationen zur Beurteilung vorgelegt werden.
Neben diesen nichtmetrischen Verfahren existieren auch drei metrische Verfahren, “Rating-Skala”, “Dollar-Metrik” oder “Konstant-Summen-Skala”, zur Einschätzung der Präferenzen. Die Details für diese Verfahren werden hier nicht mehr erläutert, weil sie für diese Diplomarbeit irrelevant sind.

Schätzung der Nutzenwerte

Bevor die Conjoint-Analyse durchgeführt werden kann, müssen Teilnutzwerte für alle Eigenschaftsausprägungen ermittelt werden. Mit Hilfe von Teilnutzwerte können metrische Gesamtnutzenwerte für alle Stimuli und die relativen Wichtigkeiten für die einzelnen Eigenschaften abgeleitet werden. [1]

Aggregation der Nutzenwerte

Nach der Schätzung der Nutzenwerte stehen die Ergebnisse der einzelnen Probanden zur Verfügung. Die Unternehmen haben in der Regel keine Interesse an der Einschätzung der einzelnen Konsumenten haben, sondern an der Einschätzung der Konsumentengruppen. Deshalb müssen diese individuellen Ergebnisse verdichtet und zu Gruppen zusammengefasst werden (Vgl.[THH00, Seite 500]). Allerdings können bei diesem Verfahren insbesondere bei heterogenen Gruppen wesentliche Informationen verloren gehen.
Auf die Details dieser Schätzung wird nicht eingegangen, weil das den Rahmen dieser Arbeit

[1] für detaillierte Darstellung dieses Schrittes siehe. [BEPW05, Seite 571-580]

sprengen würde.
Wie jede Analyse weist auch die Conjoint-Analyse Vor- und Nachteile auf.

2.1.1 Vorteile der Conjoint-Analyse

Zudem bietet sich die Conjoint-Analyse für Visualisierungen an. Ein weiterer Vorteil der Conjoint-Analyse besteht darin, dass sie mit den Resultaten der durchgeführten Marktforschung, Cluster gebildet werden können, die eine direkte Nutzenmessung beim Konsumenten erlaubt und somit ein Instrumentarium zur Zusammenfassung von Personengruppen, welche dieselben Präferenzen haben realisieren kann.
In eine Conjoint-Analyse kann sowohl subjektive als auch objektive Merkmale einbezogen werden [WWS05, Seite 119].

2.1.2 Nachteile der Conjoint-Analyse

Die Conjoint-Analyse ist sehr zeit- und kostenaufwendig, dies könnte gegen einen Einsatz der Conjoint-Analyse sprechen. Ebenso setzt sie komplexes statistisches sowie mathematisches Fachwissen voraus. Innerhalb der Conjoint-Analyse ist die Profilmethode nur für einfache Produkte mit wenig Eigenschaftsausprägungen anwendbar. Die Tatsache, dass die Eigenschaften vor der Befragung feststehen und whärend der Befragung nicht mehr geändert werden, kann auch gegen die Anwendung der Conjoint-Analyse sprechen.

2.2 Evolutionäre Algorithmen

In vorherigem Abschnitt wurde die Marktforschungsmethode “Conjoint-Analyse” kurz vorgestellt. Diese Arbeit untersucht den Ansatz, die Evolutionäre Algorithmen mit Conjoint-Analyse zu kombinieren. Im Folgenden wird versucht das Konzept evolutionärer Algorithmen zu erläutern und deren Vor- und Nachteile aufzuzeigen.
Die Algorithmen, die zu den stochastischen Such-/Optimierungsmethoden gehören und die wesentlicher Mechanismen der biologischen Evolution nachahmen, werden unter dem Begriff „Evolutionäre Algorithmen“(EA) zusammengefasst (Vgl. [NHB+98, Seite 55] und [Nis94, Seite 13]).
Um evolutionäre Algorithmen besser zu verstehen, ist es sinnvoll, dass deren Vorbild aus der Natur näher angesehen wird. Zu diesem Zweck soll in diesem Unterkapitel ein kurzer Überblick über die Prozesse der natürlichen Evolution gegeben werden. Dabei werden nur die Prozesse, die von den EA imitiert werden, kurz vorgestellt.

2.2.1 Die biologische Evolution

Die Theorie der biologischen Evolution, die von Charles Darwin (1809-1882) und von Alfred Wallace (1823-1913) zum ersten Mal repräsentiert wurde, besagt im wesentlichen zusammengefasst:

> *„As many more individuals of each species are born than can possibly survive; and as, consequently, there is a frequently recurring struggle for existence, it follows that any being, if it vary however slightly in any manner profitable to itself, under the complex and sometimes varying conditions of life, will have a better chance of surviving, and thus be naturally selected. From the strong principle of inheritance, any selected variety will tend to propagate its new and modified form.“[Dar59, Seite 5]*

Diese Schlussfolgerung basiert auf drei Beobachtungen und Annahmen.[2] Die erste Beobachtung war, trotzt eines potenziellen Überschusses der Lebewesen in der Natur, bleibt die Populationsgröße relativ konstant.
Diese Beobachtung hat Darwin interpretiert, indem er zu der Feststellung gekommen ist, dass fast alle Lebewesen mehr Nachkommen erzeugen als überleben. Demnach stirbt der größere Teil, bevor er selbst Nachkommen erzeugen kann. Die Annahme, dass es immer Unterschiede - auch wenn nur feine - zwischen den Individuen einer Art gibt, beruhte auf folgende Beobachtung: Die Lebewesen einer Art ähneln sich zwar unterschiedlich stark aber sie sind nie identisch. Deshalb ist jede Art mit einer mehr oder weniger starken Variationsbreite ausgestattet.
Erbliche Varianten, die sich im Kampf ums Überleben bewährt haben, bevorzugt in den Folgegenerationen wiederfinden. So können sich die kleinen Variationen der Individuen einer Art quasi addieren und somit im Laufe der Generationen zur Vollkommenheit und Optimierung von Organen und Eigenschaften der Lebewesen führen. Dies stellte die dritte Beobachtung von Darwin dar.
Darwin beschrieb die Evolution als einen stufenweisen Prozess, dessen Zentrum „die Mutation“ und „die natürliche Selektion“ bildet und bei dem nur die Bestangepassten eine höhere Chance haben *„ survival of the fittest“* , ihre Erbanlagen weiterzugeben. Die Evolution hat folgende drei wesentliche Faktoren, die in den Optimierungsverfahren auf einer abstrakter Ebene imitiert werden:

- Mutation
- Selektion
- Rekombination

Mutation

Spontane Veränderungen im Genotyp[3] werden als Mutationen bezeichnet. Die Mutationen können in verschiedenen Formen wie Genmutation oder Chromosomenmutation auftreten und werden von unterschiedlichen Faktoren wie Temperatur oder ionisierende Strahlung beeinflusst.
Die Änderungen des Phänotypus[4] eines Individuums werden durch Mutation hervorgerufen. Eine sehr kleine Veränderung eines Gens kann sehr starke Auswirkungen auf Phänotyp haben.
Mutationen bilden die Basis für die Veränderungen in der Evolution. In der Regel finden große

[2] Die nachfolgende Ausführungen stützen sich insbesondere auf die Darstellungen bei[SHF94, Seite 40-42]
[3] Der Genotyp oder das Erbbild eines Organismus repräsentiert seine exakte genetische Ausstattung, also den individuellen Satz von Genen, den er im Zellkern in sich trägt
[4] Das Erscheinungsbild eines Lebewesens

Veränderungen in einer Population stufenweise durch Addition von vielen kleinen Mutationen statt.

Selektion

Ausgehend von der Mutation, wird bei der Selektion diejenige Gene ausgewählt, die dem Träger dabei helfen, sich optimal an ihren Umgebung anzupassen. Das Resultat dieser Anpassung, welches durch die Selektion zustande gekommen ist, verleiht den Individuen die Fähigkeit, ihre Erbanlagen an die folgende Generation weiterzugeben.

Diese Fähigkeit wird als reproduktive *Fitness* bezeichnet. Die relative Fitness eines Individuums wird wie folgt definiert:

$$W_X = \frac{Nachkommenschaft\ des\ Genotyps\ x}{Nachkommenschaft\ des\ bestenGenotyps}$$

Die Selektion in der Natur ist ein stochastischer Prozess, weshalb auch die weniger angepassten Individuen, auch wenn nur gering, die Chance haben, ihre Erbanlagen weiterzugeben.
Für eine unterschiedliche Tauglichkeit und Reproduktivität können folgende Ursachen genannt werden. Vgl. [Wei02, Seite 29-30]

- Unterschiedliche Überlebenschancen, z.B. in der Lebensfähigkeit oder dem Behauptungsvermögen gegen Rivalen.
- Unterschiedliche Fruchtbarkeit bzw. Fortpflanzungsraten.
- Unterschiedliche Fähigkeit einen Geschlechtspartner zu finden.
- Unterschiedliche Länge der Generationsdauer.

Der Selektionprozess stellt den einzigen gerichteten Prozess in der Evolution, dessen Ziel die Anpassung an veränderte Bedingungen ist. Dieser Prozess schützt auch die Harmonie zwischen Gene eines Genpools, indem er die disharmonischen Kombinationen, die durch Mutationen auftreten, eliminiert, bis ein veränderter Zustand erreicht ist. Dies wird als *Stabilisierende Selektion* bezeichnet.

Rekombination

Als genetische Rekombination bezeichnet man die Entstehung neuer Kombinationen von Genen durch geschlechtliche Fortpflanzung. Bei der Rekombination werden keine Neuigkeiten eingeführt, sondern nur Vorhandenes zusammengestellt. Deshalb ist die Rekombination aus Sicht der klassischen Evolutionslehre kein Evolutionsfaktor. Nach den jüngeren Forschungsergebnissen allerdings wird angenommen, dass eine sehr starke Vernetzung der Genen in den genotypischen Strukturen für phänotypische Zusammenhänge verantwortlich sind, weshalb auch die Rekombination zu den Evolutionsfaktoren zählt.[Wei02]

2.2.2 Hauptformen Evolutionärer Algorithmen

Die Evolutionäre Algorithmen unterscheiden sich von konventionellen Such- und Optimierungsverfahren u. a. mit der biologischen Evolution entlehnte Terminologie. Die Tabelle 2.1 gibt einen Überblick über die Begriffe der biologischen Evolution und deren Benutzung in den Evolutionären Algorithmen.

Ausdruck	**Bedeutung**
Individuum	Struktur (enthält in geeigneter Weise repräsentierten Elemente einer Lösung)
Population (von Individuen)	Menge von Strukturen (Lösungsalternativen)
Eltern	zur Reproduktion ausgewählte Lösungen Individuen
Kinder, Nachkommen	aus den Eltern erzeugte Lösungen
Crossover	Suchoperator, der Elemente mehrerer Individuen vermischt
Mutation	Suchoperator, der ein Individuum modifiziert
Fitness	Lösungsqualität hinsichtlich der relevanten Zielkriterien
Generation	Verfahrensiteration

Nur bei Genetischen Algorithmen gebräuchlich:

Chromosom (besteht aus Genen)	Darstellungsform einer vollständigen Lösung. Grundsätzlich identisch mit Individuum; gelegentlich kann ein Individuum sich aus mehreren Chromosomen zusammensetzen; übliche Form: String
Gen	Bit (binäre Codierung unterstellt)
Allel	Ausprägung eines Bits/Gens (binär: 0 oder 1)
Genotyp	Codierte Lösung (String)
Phänotyp	Decodierte Lösung

Tabelle 2.1: EA-Fachterminologie

Quelle : [NHB+98] und [Nis94]

Die EA imitieren unterschiedlich stark verschiedene Aspekte der biologischen Evolution. Das führt dazu, dass die Evolutionären Algorithmen im Wesentlichen in vier Hauptformen unterteilt werden (Abbildung 2.1).
Die Algorithmen, die besonders genetische Mechanismen auf Chromosomen-Ebene betonen, werden als Genetische Algorithmen (GA) bzw. Genetische Programmierung (GP) bezeichnet. Evolutionsstrategien (ES) nehmen die phänotypische Ebene der biologischen Evolution als Vorbild. Evolutionäre Programmierung (EP) betrachtet dagegen die biologische Evolution auf der Ebene der ganzen Populationen. Zu Beginn wurden die vier Hauptformen der Evolutionären Algorithmen als nebeneinander existierte konkurrierende Modelle gesehen.
Als Oberbegriff wurde *„Evolutionäre Algorithmen“* erst später, nachdem alle Formen Erfolge in

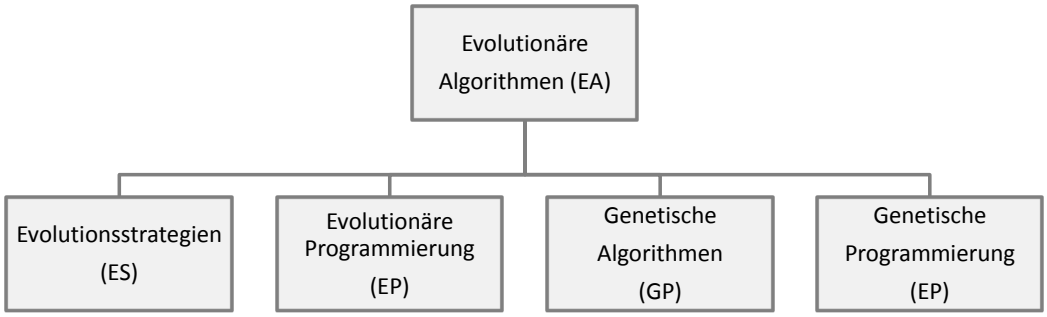


Abbildung 2.1: Hauptformen der Evolutionären Algorithmen

den Anwendungen vorgewiesen haben, benutzt.
Alle evolutionären Algorithmen sind strukturell gleich aufgebaut. Der Grundbau eines Evolutionären Algorithmus kann wie folgt beschrieben werden.
Vorerst ist ein Suchraum zu bestimmen, der die prinzipielle Struktur der Lösung darstellt, unabhängig von der Qualität der Aussage, ob die Lösung gut oder schlecht ist. Damit die Komponenten des Suchraumes evaluiert und moduliert werden können, ist eine Zielfunktion notwendig, um abschätzen zu können, wie gut oder schlecht eine Lösung ist. EA konstruieren daraufhin eine Startpopulation von überwiegend willkürlichen Lösungen. Daraufhin werden von der Startpopulation aus neue Generationen von Lösungen generiert, indem die genetischen Operationen wie die *Mutation*, *Rekombination* und die *Selektion* eingesetzt werden. Dieser Vorgang wird wiederholt, bis eine ausreichend gute Lösung gefunden wird oder die Population konvergiert ist (Abbildung 2.2).

1 **Vorbedingungen:** $\mu = x$ `{*Populationanzahl=x *}`
2 Wähle Strategieparameterwerte
3 Initialisiere Ausgangspopulation $P(0)$
4 $t \leftarrow 0$
5 Bewerte Individuen der Ausgangspopulation
6 **wiederhole** `{*Generationszyklus*}`
7 $t \leftarrow t + 1$
8 Selektion `{*Auswahl der Eltern*}`
9 Replikation `{*Nachkommen`
10 Variation `generieren*}`
11 Bewerten der Nachkommen
12 Bilden der neuen Population $P(t)$
13 **bis** $tmax$ *erreicht ist oder eine andere Abbruchbedingung erfüllt ist*
14 Ausgabe der Ergebnisse
15 **Stop**
16 **Nachbedingungen:** $\mu = x$

Abbildung 2.2: Allgemeines Ablaufschema eines EA in Pseudocode

Quelle : [Nis97, Seite 15]

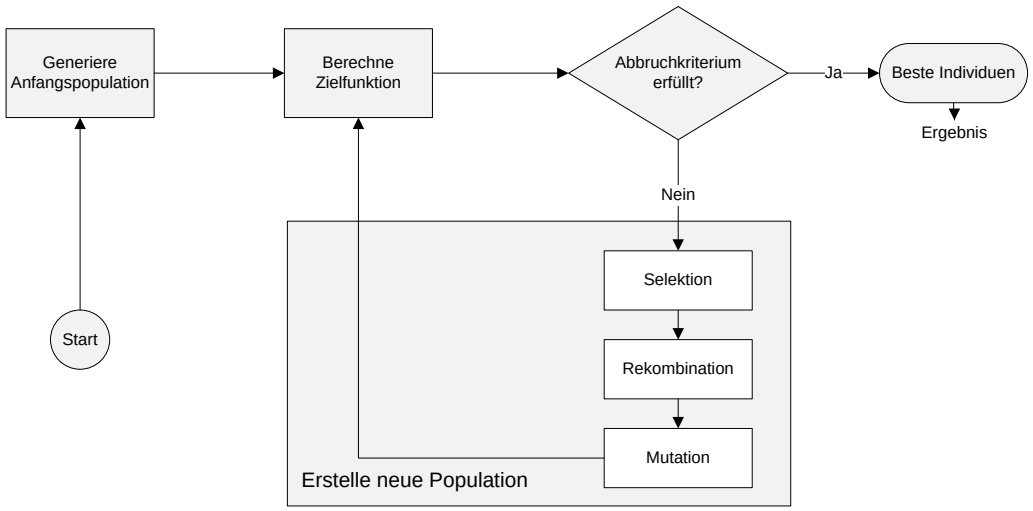


Abbildung 2.3: Ablauf eines Standard-EA

2.2.2.1 Evolutionsstrategien

Eine der ersten Hauptformen der Evolutionären Algorithmen sind die Evolutionsstrategien.

Sie basieren auf einem Modell der Evolution, das in den sechziger Jahren von Ingo Rechenberg, der gegenwärtig an der Technischen Universität Berlin lehrt, für Optimierungsprobleme aus dem technisch-physikalischen Bereich entwickelt. Dieses Modell wurde von ihm selbst und von den anderen Forschern, insbesondere von Hans-Paul Schwefel weiterentwickelt. Der Grundgedanke, die biologische Evolution als Vorbild für rechnerische Lösungsversuche anzuwenden, resultierte aus einem praktischen Problem, nämlich den Luftwiderstand eines Tragflügels zu minimieren. Deshalb sind sie auf Optimierungsprobleme mit kontinuierlichen Parametern zugeschnitten[uE]. Abbildung 2.3 stellt die Struktur eines einfachen Evolutionären Algorithmus grafisch dar. Im Folgenden wird anhand der Abbildung 2.3 versucht das Grundkonzept über den Ablauf der EA bzw. ES darzustellen.

Die ES haben ihre Ursprünge in Optimierungsproblemen. Deshalb wird versucht, eine reelle Funktion $f : R^n \rightarrow R$ mit n kontinuierlichen Parametern zu optimieren, wobei die Funktion f auch auf eine Teilmenge des $f : R^n$ definiert werden darf. (Vgl. [GKK04, Seite 155])
Die Grundlage der ES bildet eine Population von μ Individuen. Jedes Individuum $\vec{a}$ enthält, als Vektor stetiger Größen $\vec{x} = x_1, x_2, \ldots, x_m$, Werte für alle m Entscheidungsvariablen des betrachteten Optimierungsproblems und sogenannte *Mutationsschrittweiten* σ_w, wobei $w \leq m$ ist. Vgl. [Nis94, Seite 141]

1. Schritt: Initialisierung der Startpopulation

Falls kein entsprechendes Vorwissen über das globale Optimum vorhanden ist, wird die Startpopulation $P(t = 0)$ mit der Größe μ zufällig und möglichst gleichmäßig erzeugt, sonst werden

einige Entscheidungsvariablen gezielt im Suchraum der Lösung verteilt. Dadurch steigt die Chance, dass die Ausgangslösung in der Nähe des globalen Optimums liegt und damit die Suche eher erfolgreich beendet werden kann.

2. Schritt: Berechnen der Zielfunktion

Die Zielfunktion $f : R^n \rightarrow R$ wird auf jedes Individuum $\vec{a}_i$ aus der Startpopulation bzw. auf die Nachkommen angewandt und der Funktionswert bestimmt. Der errechnete Funktionswert ist in der Regel der Fitnesswert eines Individuums, der die Basis für die Akzeptanz einer Lösung darstellt. Nach diesem Schritt wird entschieden, ob die Lösungssuche beendet werden kann oder die folgenden Schritte wiederholt werden müssen.

3. Schritt: Erzeugen der Nachkommen

Aus μ Eltern werden λ Nachkommen erzeugt. Traditionell wird ein Verhältnis μ/λ von ungefähr 1/7 empfohlen. Standardwerte sind hierfür $\mu = 15$ und $\lambda = 100$. Vgl. [Nis97, Seite 157]
Der Quotient $s = \mu/\lambda$ bestimmt den Selektionsdruck, wobei $0 \leq s \leq 1$ ist.
Das Erzeugen der Nachkommen erfolgt durch die Iteration folgender drei Teilschritten: *Selektion, Rekombination* und *Mutation*.

Teilschritt 3-1: Selektion

Wie bereits unter 2.1.1 erwähnt, stellt die Selektion den einzigen gerichteten Prozess in der biologischen Evolution dar. Entsprechend dazu hat die Selektion in den ES die Aufgabe, die Suche schrittweise in eine Richtung zu leiten. Dadurch wird gewährleistet, dass die Suche in den relevanten Bereichen des Suchraumes stattfindet.
Dieser Operator kann sowohl auf Eltern, um zu bestimmen welche Individuen ein Nachkomme erzeugen dürfen, als auch auf Nachkommen, um zu bestimmen welche die erzeugten Nachkommen als Eltern für nächste Population übernehmen werden dürfen, angewandt werden.
In der Grundform der ES wird dieser Operatorjedoch nur auf Eltern angewandt, um aus jeweils zwei stochastisch ausgewählten Individuen (E_1, E_2) ein Nachkomme (K) zu erzeugen. Die Selektionswahrscheinlichkeit der Individuen in der Grundform ist gleich und beträgt $1/\mu$.
Das Ergebnis der Selektion wird als Eingabeparameter an den Rekombinationsoperator übergeben.

Teilschritt 3-2: Rekombination

Die Rekombination hat die Aufgabe die Werte der Entscheidungsvariablen des erzeugten Nachkommens aus den entsprechenden Variablen der Eltern zusammenzusetzen. In der Regel kommen in der Grundform der ES zwei Rekombinationsmöglichkeiten, die *diskrete Rekombination* und die *intermediäre Rekombination*, vor.
Bei der diskreten Rekombination wird für jede Variable des Kindes entschieden, welcher Variablenwert von welchem Elter übernommen werden soll. Die Werte des Kindes sind somit eine Teilmenge der Eltern Variablenwerte.

Die intermediäre Rekombination bildet dagegen für jeden Variablenwert des Kindes einen Durchschnittswert aus der elterlichen Variable. Tabelle 2.2 veranschaulicht die zwei Rekombinationsformen.
Bezüglich der Wichtigkeit der Rekombination in den ES sind in der Literatur entgegengesetzte

diskrete Rekombination

Elter 1	4,0	6,2	1,8	0,3
Elter 2	8,0	3,6	0,8	0,7
Nachkomme	4,0	3,6	0,8	0,3

intermediäre Rekombination

Elter 1	4,0	6,2	1,8	0,3
Elter 2	8,0	3,6	0,8	0,7
Nachkomme	6,0	4,9	1,3	0,5

Tabelle 2.2: diskrete & intermediäre Rekombination

Meinungen vertreten. Nissen [Nis97, Seite 158] bezeichnet diesen Operator als wichtig für die Beschleunigung des Optimierungsprozesses. Weicker [Wei02, Seite 132] bezeichnet dagegen den Rekombinationsoperator als Hintergrundoperator.

Teilschritt 3-3: Mutation

In diesem Prozess wird der erzeugte Nachkomme (K), wie in der biologischen Mutation, mit kleinen Schritten zufällig verändert. Dies geschieht mit Hilfe von Mutationsschrittweiten (σ). Zunächst werden die Mutationsschrittweiten durch Multiplikation mit einer logarithmisch normalverteilten Zufallsgröße verändert. Anschließend werden alle Entscheidungsvariablen (n) des Individuums mutiert, indem die Variablenwerte mit einer Zufallsgröße, die einen Erwartungswert 0 und eine Standardabweichung von σ hat, addiert werden. Formal sieht dieser Schritt wie folgt aus: Vgl. [Nis97, Seite 159])

Standardabweichung:

Falls nur mit einer Standardabweichung gearbeitet wird :

$$\acute{\sigma} = \sigma.exp(\tau_0.N(0,1))$$

Für jedes Kind (K) *eine separate Standardabweichung:*

$$\acute{\sigma_k} = \sigma_k.exp(\tau_1.N(0,1) + \tau_2.N_k(0,1))$$

Die Strategieparameter τ_1 *und* τ_2 *sind dabei exogene Konstante.*[4]

Mutation der Kindesvariablen:

$$\acute{x_j} = x_j + \acute{\sigma}.N_j(0,1) \qquad j = (1,2,..,n) \text{ und } (\forall \geq n_\sigma : \acute{\sigma_j} = \acute{\sigma_{n_\sigma}})^5$$

[4] Optimale Werte sind hierfür, nach Nissen 0,1 bis 0,2 [Nis97, Seite 159]
[5] Dies gilt nur, wenn für jedes Kind eine separate Standardabweichung eingesetzt wird.

Die standard-normalverteilte Zufallszahl ($N(0,1)$) passt die globalen Mutationsschrittweiten, ($N_k(0,1)$) dagegen die individuellen Mutationsschrittweiten jedes Individuums an. Dadurch ist ES in der Lage die Werte für die Standardabweichung im Laufe der Zeit automatisch anzupassen, was zu einer großen Flexibilität des Verfahrens führt. Anschließend wird der *2. Schritt* und - wenn nötig- der *3. Schritt* mit Teilschritten λ-mal wiederholt.

Derzeit existiert eine Vielzahl von Evolutionsstrategien, die die biologische Evolution in unterschiedliche Komplexitätsstufen imitiert. Sie sind in sich homogen und partiell aufeinander abgestimmt. Sie differenzieren sich hauptsächlich in der Art und Weise, wie Population und Individuen mutiert, rekombiniert, resümiert und selektiert werden.
Diese Varianten der ES werden in den nächsten Abschnitten kurz beschrieben.

(1 + 1) Evolutionsstrategie

Die sogenannte „zweigliedrige“ $(1 + 1)$ Evolutionsstrategie ist die einfachste Evolutionsstrategie. Sie wurde bereits im Jahre 1964 von Rechenberg bei „Darwin-im-Windkanal-Experiment“, in dem eine zur Zickzackform gefaltete Gelenkplatte sich evolutiv zur ebenen Form geringsten Widerstands entwickelt, eingesetzt.[Rec]
Bei dieser Variante wird von einem Elter, der durch einen Vektor der Reelenzahlen repräsentiert wird, ein Nachkomme erzeugt. Die Erzeugung des Nachkommens erfolgt bei dieser Variante durch einfaches Duplizieren des Ausgangsvektors.
Anschließend wird der Nachkomme mutiert, in dem auf jeden einzelnen Zahlenwert des Vektors ein zufälliger positiver oder negativer Wert addiert wird.
Im nächsten Schritt werden der Elter und der Nachkomme durch die Zielfunktion bewertet.
Die drei wesentlichen Schritte: *Duplikation, Mutation* und *Selektion* werden solange wiederholt, bis eine ausreichend gute Lösung gefunden wird oder eine Abbruchbedingung erfüllt ist. Diese Variante verarbeitet grundsätzlich ein einziges Individuum, nur bis Selektion existieren kurzzeitig zwei Individuen und läuft seriell ab.
Der Name der $(1 + 1)$-ES stammt aus dem Ablauf und dem Konzept der Strategie: die erste „1“ symbolisiert den Eltern und die zweite „1“ den Nachkommen. „+“ bedeutet, dass der Elter und der Nachkomme zusammen durch die Zielfunktion bewertet werden.

$(\mu + \lambda)$ Evolutionsstrategie

Die $(\mu + \lambda)$-ES ist eine Verallgemeinerung der $(1 + 1)$-ES. Der Nachteil der $(1 + 1)$-ES, dass sie seriell abläuft, wird mit Hilfe von dieser Verallgemeinerung überwunden. Dazu wird eine Population mit der Größe (μ) anstelle eines Individuums bearbeitet, aus dieser Population werden später λ Nachkommen erzeugt. (μ) und (λ) sind beliebige ganze Zahlen, die so zu wählen sind, dass $\lambda \geq \mu \geq 1$ ist. Vgl. [SHF94, Seite 152] Die Erzeugung der λ Nachkommen erfolgt dadurch, dass aus μ Eltern-Individuen λ Eltern zufällig ausgewählt und wie bei $(1 + 1)$-ES einfach dupliziert werden. Infolge der eingesetzten statischen Gleichverteilung trifft die Auswahl der λ Eltern aus der μ Eltern mit gleicher Wahrscheinlichkeit. Dabei ist eine Mehrfachauswahl möglich und insbesondere im Falle $\mu < \lambda$ nötig.
Die anderen Schritte dieser Variante ähneln sich im Wesentlichen der $(1 + 1)$-ES.

In dieser Variante wird die Zielfunktion sowohl auf Eltern-Individuen als auch auf Kinder-Individuen angewandt, was zur Folge hat, dass immer die besten Individuen überleben und die Qualität der Generationen nie schlechter wird. Dies hat aber den Nachteil, dass die Suche vorzeitig gegen das lokale Optimum konvergiert.

(μ, λ) Evolutionsstrategie

Die Tatsache, dass bei der $(\mu + \lambda)$-Variante die Eltern und die Kinder zusammen bewertet werden und nur die besten μ Individuen überleben, hat einige Nachteile. So können z. B. sogenannte „unsterbliche Individuen“ entstehen, dank ihren besseren Fitnesswerten mehrere Generationen überleben. Dies hat u. a. zur Folge, dass die Suche vorzeitig gegen das lokale Optimum konvergiert, wenn das sogenannte unsterbliche Individuum ein lokales Optimum der Zielfunktion ist. Die Suche wird von dem lokalen Optimum auf einen falschen Suchpfad geleitet, weshalb das globale Optimum meistens nicht gefunden werden kann.
(μ, λ)-ES wurde im Jahre 1975 von Hans-Paul Schwefel in seiner Promotion formuliert [uE]. Diese Variante vermeidet die vorzeitige Konvergenz der $(\mu + \lambda)$-Variante, indem sie den Selektionsmechanismus der $(\mu + \lambda)$-ES verändert. Im Gegensatz zur $(\mu + \lambda)$-ES werden die besten Individuen aus der Vorgängergeneration nicht zur Selektion herangezogen, stattdessen selektiert die (μ, λ)-ES die μ Eltern-Individuen für die nächste Generation nur aus den λ Nachkommen. Je nachdem, ob die Zielfunktion maximiert oder minimiert werden soll, verhält sich der Verlauf der Zielfunktion bei der $(\mu+\lambda)$-ES monoton steigend oder fallend. Die Zielfunktion der (μ, λ)-ES verhält sich dagegen in der Regel stark nach oben und nach unten schwankend. Abbildung 2.4 veranschaulicht den Verlauf der Qualität der Individuen beider ES-Varianten.
Die anderen Schritte dieser Variante entsprechen im Wesentlichen der $(\mu + \lambda)$-ES. (μ, λ)-ES und $(\mu + \lambda)$-ES werden, falls die Selektionsstrategie keine Rolle spielt, zur $(\mu \# \lambda)$-ES zusammengefasst, wobei „$\#$“ für „$+$“ oder „ , “ steht.

$(\mu/\rho \# \lambda)$ Evolutionsstrategie

Im Gegensatz zu den oben genannten ES verwenden die $(\mu/\rho \# \lambda)$-ES die sexuellen Rekombinationen. Diese Variante läuft im Prinzip genau so ab, wie die $(\mu \# \lambda)$ bis auf die Erzeugung der Nachkommen. Bei der $(\mu/\rho \# \lambda)$-ES wird die Rekombination benutzt, um die Nachkommen zu erzeugen, statt einfaches Duplizieren der Eltern. Dazu werden λ Gruppen bestehend aus ρ Elter-Individuen zufällig mit gleicher Wahrscheinlichkeit ausgewählt. In der Regel wird eine Gruppe bestehend aus zwei Elter-Individuen ($\rho = 2$) zur Erzeugung ausgewählt. Die Variablenwerte der Nachkommen werden entweder mit Hilfe von einer der oben genannten Rekombinationsmöglichkeiten zusammengesetzt. Im weiteren Strategieverlauf werden die entsprechenden Schritte der bereits angewandten $(\mu \# \lambda)$-ES eingesetzt.
Neben den oben genannten Varianten, die auf der Basis einzelner Individuen arbeiten, existieren weitere Strategien, die auf Basis ganzer Populationen arbeiten und damit eine parallele Verarbeitung der ES ermöglichen. Auf die Einzelheiten der Population basierten Strategien und auf die Erweiterungen der Standard-ES wird aber nicht eingegangen, weil das den Rahmen dieser Arbeit sprengen würde.

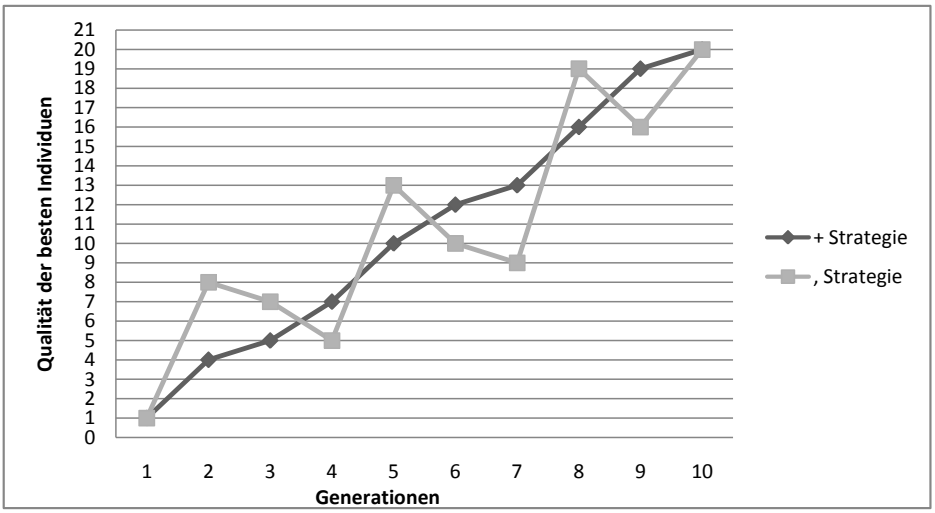


Abbildung 2.4: Qualität der besten Individuen

2.2.2.2 Genetische Algorithmen

Die GA haben ihren Ursprung in den Arbeiten von John Holland, in denen er die Mechanismen der adaptiven Systeme Mitte der 60er Jahre zu erklären versuchte.
John Holland hat die biologische Evolution aus einem anderen Blickwinkel betrachtet als Ingo Rechenberg.
Während Rechenberg die phänotypische Ebene der biologischen Evolution als Vorbild für seine Theorien nahm, hat sich John Holland überwiegend für die genotypische Ebene der biologischen Evolution interessiert.

> *„Er stellte sich die Frage, wie es die Natur fertig bringt, mit Hilfe des genetischen Codes und der genetischen Prozesse so etwas Erstaunliches wie Intelligenz, Selbstorganisation und die komplexesten Formen der Adaption hervorzubringen, und wie man dies auf künstlichem Wege nachbauen und nutzen kann.“ Vgl.[SHF94, Seite 185]*

Anders als EA sind die GA nicht von Anfang an für die Optimierungsprozesse konzipiert worden. Erst durch die Arbeiten von De Jong im Jahre 1975 wurden die GA für Optimierungsprobleme angewandt.
Da es für die Genetische Algorithmen keine formale Beschreibung gibt, wird die Struktur eines

GA häufig in eine Art Pseudocodes formuliert. Die unterschiedlichen Varianten, welche im Laufe der Zeit entwickelt worden sind, unterscheiden sich hauptsächlich in der Codierung und den verwendeten Verfahren.
Im nächsten Abschnitt wird versucht einen kurzen Überblick über das Grundkonzept einer GA zu verschaffen [7].
Wie bereits erwähnt, imitieren die GA der biologischen Evolution auf der genotypischen Ebene, deshalb sind die Individuen einer GA-Population -anders als eine EA-Population- die künstlichen „Chromosomen“ im weiterem Verlauf „Strings“ oder „Individuen“ genannt. Jedes Individuum $\vec{a}$ wird durch einen String mit der Länge (L) repräsentiert. Die Länge L ist anwendungsabhängig.

Die Variablen des Optimierungsproblems werden mit Hilfe von Segmenten (n und $n \leq L$), die aus unterschiedlich langen Bitfolgen bestehen, realisiert. Die einzelne Bits werden als „Gene“ und deren Ausprägungen (0 oder 1) als „Allel“ bezeichnet[8]. Abbildung 2.5 veranschaulicht die binäre Lösungscodierung bei GA.

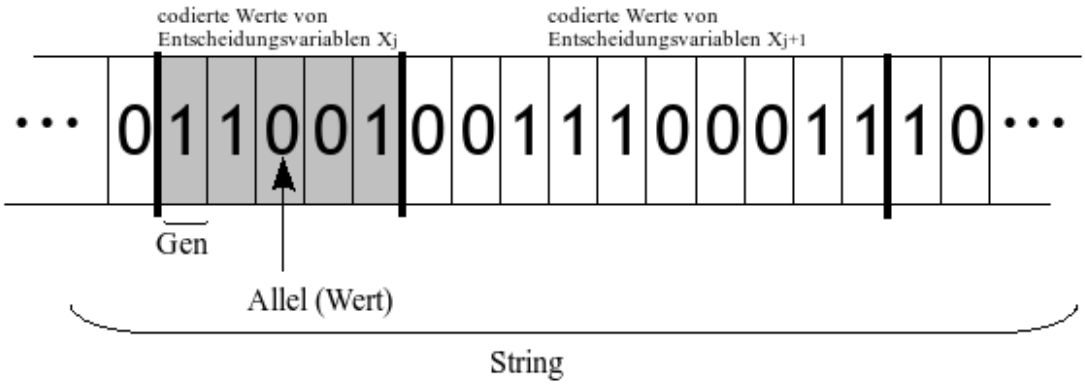


Abbildung 2.5: Binäre Lösungscodierung bei GA

Quelle : [Nis97, Seite 35]

Der Ablauf eines GA kann in die folgenden Hauptschritte unterteilt werden:

- Initialisierung
- Bewerten der Ausgangslösungen
- Stochastische Selektion und Replikation
- Erzeugen der Nachkommen
- Prüfen der Abbruchkriterien und ggf. Wiederholung der Schritte 3 bis 5

Im Folgenden werden diese Schritte kurz erklärt, dabei wird der Fall betrachtet, dass die Zielfunktion $F(\vec{x})$ mit n Entscheidungsvariablen $\vec{x} = x_1, x_2, \ldots x_n$ maximiert werden soll.

[7] Die nachfolgende Ausführungen stützen sich insbesondere auf die Darstellungen bei [Nis97, Seite 34-41]
[8] Vgl. Tabelle 2.1

1. Schritt: Initialisierung

Die Ausgangspopulation $P(t = 0)$ von μ Individuen wird in der Regel zufällig erzeugt. Der Wert von μ ist grundsätzlich eine gerade Zahl. Dann werden die Individuen binär codiert, indem die Allele mit gleicher Wahrscheinlichkeit entweder auf „Null“ oder auf „Eins“ gesetzt werden.
Abhängig von der Problemstellung kann es vorkommen, dass die Startpopulation mit Heuristiken beeinflusst wird und die Chromosomen mit großer Fitness in die Startpopulation herangezogen werden. Das bringt aber die Gefahr mit sich, dass diese Chromosomen sich sehr schnell durchsetzen und die Suche vorzeitig gegen das lokale Optimum konvergiert.

2. Schritt: Bewerten der Ausgangslösungen

Dieser Schritt differenziert die Lösungsalternativen für darauf aufbauenden Selektionsschritt nach Qualität. Die Individuen der Startpopulation werden decodiert und mit Hilfe von Fitnessfunktion Φ, die sich aus der Zielfunktion und Decodierfunktion zusammensetzt, bewertet. Weil in GA generell das Prinzip „die Stärkste überleben (*survival of fittest*)“ gilt, ist der Wert dieser Funktion für das Überleben eines Chromosoms sehr wichtig und stellt die Überlebenswahrscheinlichkeit eines Individuums dar.

3. Schritt: Stochastische Selektion und Replikation

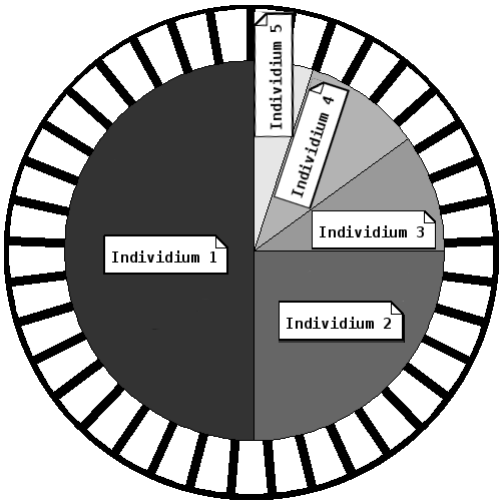
In diesem Schritt wird anhand der Selektionswahrscheinlichkeit Ps bestimmt, welche Individuen als Eltern für die Erzeugung der Nachkommen übernommen werden. Die Ziehung findet „mit Zurücklegen“ statt, so dass die Duplikate erlaubt sind. Die Selektionswahrscheinlichkeit Ps eines Individuums $\vec{a_i} \in P$ ergibt sich wie folgt Vgl.[Nis97, Seite 37]:

$$Ps(\vec{a_i}) = \frac{\Phi(\vec{a_i})}{\sum_{j=1}^{\mu} \Phi(\vec{a_i})}$$

Bei diesem Selektionsverfahren hat jedes Individuum eine positive Selektionswahrscheinlichkeit und damit auch die Möglichkeit als Eltern ausgewählt zu werden. Diese Art der Selektion wird als „Rouletterad-Selektion“ genannt.
Bei der „Rouletterad-Selektion“ wird jedem Individuum ein Abschnitt proportional zu seiner Selektionswahrscheinlichkeit auf einem virtuellen Rad zugeordnet. Bei der Auswahl der Eltern der Folgegeneration wird durch eine gleichverteilte Zufallszahl zwischen 0 und 1 zufällig ein Abschnitt des „Rads“ ausgewählt und das entsprechende Individuum wird Elter übernommen. Der ausgewählte Elter wird in die Zwischenpopulation kopiert. Dieser Vorgang wird μ-mal wiederholt. Abbildung 2.6 veranschaulicht die „Rouletterad-Selektion“.

4. Schritt: Erzeugen der Nachkommen

Dieser Schritt besteht aus vier Teilschritten, die $\mu/2$-mal durchlaufen müssen.



Individuum	Sel. Wahrsch.
1	0,5
2	0,25
3	0,1
4	0,1
5	0,05

Abbildung 2.6: Roulletterad-Selektion

Teilschritt 4-1: Stochastische Partnerauswahl
Aus der Zwischenpopulation werden mit der Wahrscheinlichkeit $1/\mu$ ohne Zurücklegen zwei Eltern ($\vec{a_{E_1}}$ und $\vec{a_{E_2}}$) gezogen. Auf beide Eltern werden der Variation-Operator und der Mutation-Operator angewandt, um zwei Nachkommen zu erzeugen.

Teilschritt 4-2: Crossover
Mit Hilfe von Variation-Operator „Crossover“ ist es möglich, die problemspezifische Informationen in den Individuen abzulegen, deshalb bildet das Crossover den Hauptoperator bei GA. Die Anwendung des Crossover hängt von der a priori festgelegten Crossover-Wahrscheinlichkeit (Pc) ab, der Crossover findet nur statt, wenn $Pc \geq U$ ist, wobei U eine Variable, deren Wert im „$[0, 1[$“-Intervall liegt, ist. Es gibt mehrere Varianten dieses Operators wie z. B. „one-point-crossover“, „two-point-crossover“ und „n-point-crossover“. Die in der Grundform verwendete „one-point-crossover“ Variante läuft folgendermaßen ab: Auf Basis einer Gleichverteilung wird ein Crossover-Punkt zwischen 1 und L-1 ohne Rücksicht auf Segmente festgelegt, der auf beide Individuen gleich ist. Die Allele rechts vom Crossover-Punkt werden anschließend zwischen zwei Individuen ausgetauscht, so entstehen zwei rekombinierte Nachkommen. Mit der Übergabe der erzeugten Nachkommen bzw. der ausgewählten Eltern, falls kein Crossover stattgefunden hat, an den Mutation-Operator wird dieser Teilschritt abgeschlossen.

Teilschritt 4-3: Mutation
Mutation ist ein Hintergrundoperator, der die Aufgabe hat, die vorzeitige Konvergenz zu verhindern. Dieser Operator invertiert jedes Bit eines Individuums abhängig von dem zufälligen Wert der Variablen U , deren Wert im „$[0, 1[$“-Intervall liegt, mit einer Wahrscheinlichkeit von Pm wenn, $U_i \leq Pm, i = (1, 2, \ldots L)$ ist, sonst werden die Bits nicht invertiert.

Teilschritt 4-4: Bewerten der Nachkommen und Ergänzen der neuen Population
Die in Teilschritten von 1 bis 3 erzeugten Nachkommen werden mit Hilfe der Zielfunktion ($F(\vec{x})$) bewertet und in die nächste Population ($P(t + 1)$ übernommen, bis die neue Population die festgelegte Größe (μ) hat. Diese neu entstandene Population ersetzt dann die vorherige Population.

5. Schritt: Prüfen der Abbruchkriterien und ggf. Wiederholung der Schritte 3 bis 4

Die bereits beschriebenen Hauptschritte, einschließlich der Teilschritte , werden im Anschluss solange wiederholt, bis ein Abbruchkriterium erfüllt wird. Exemplarisch erfüllen folgende Parameter das Abbruchkriterium:

- Das Erreichen der Anzahl von maximal zu erzeugenden Generationen
- Das Finden der optimalen Lösung
- Das Verbrauchen der verfügbaren Ressourcen

Seit dem Entwurf von Holland wurde eine Vielzahl der bereits erwähnten Prinzipien in vielerlei Hinsicht weiterentwickelt. Gerade um GA auf unterschiedliche Optimierungsprobleme anwenden zu können, wurden insbesondere die Selektion und die Suchoperatoren erweitert. Da eine detaillierte Ausführung diese Themenschwerpunkte den Rahmen dieser Arbeit sprengen würde, wird auf die Erweiterungen der GA-Prinzipien nicht weiter eingegangen.

2.3 Evolutionäre Conjoint

Im Abschnitt 2.1.2 wurde einige Nachteile aufgeführt, um diese umzugehen, wurde in jüngster Zeit der Ansatz, EA mit der Conjoint-Analyse zu kombinieren, aufgegriffen.
In der Marktforschung werden EA vorwiegend in der Neuproduktentwicklung Anwendung, das Design des neuen Produktes zu optimieren. Der Erfolg dieses Ansatzes wurde in der Praxis durch von der Firma “Affinnova”[9] entwickelte IDDEA®Technologie bestätigt[aff].
Die evolutionäre Conjoint-Analyse ist stark ausgerichtet an die Methode der Interaktiven Evolutionären Algorithmen (IEA) [10].
Die IEA ähneln sich stark den aus Abschnitt 2.2 vorgestellten klassischen EA, weshalb hier nur die die Unterschiede erläutert werden. Darüber hinaus werden die Gemeinsamkeiten von IEA und Conjoint-Analyse erwähnt.
Die IEA betrachtet ein Produkt, als ein Bündel von Merkmalen. Die Merkmale können als Gene oder Chromosomen in der Genetik angesehen werden. Diese Sichtweise ähnelt sich sehr dem unter Erhebungsdesign einer Conjoint-Analyse durchzuführenden Unterschritt “Definition der Stimuli”.
Die Bewertung der gegenwärtigen Lösung bzw. Population erfolgt durch die Entscheidungen der Benutzer. Sie dient als Fitnessfunktion, dessen Qualitätsmaaß an jeder Produktalternative der gegenwärtigen Population gemessen ist. Dies entspricht dem Schritt “Bewertung der Stimuli” in der Conjoint-Analyse. Die beiden Methoden “Rangreihung” und “Profilpaarmethode” für die Bewertung der Stimuli können hier angewandt werden, aber auch die Kombination der beiden Methode kann zum Einsatz kommen.
Die Kundenpräferenzen werden bei IEA in den Optimierungsprozess einbezogen. Diese Vorgehensweise fördert eine gute Lösung der Optimierungsprobleme in hohen Dimensionslösungsräumen, weil die Fortbestand und Funktionalität des Optimierungsprozess mit Hilfe von den Kundenpräferenzen gewährleistet werden. Aufgrund der ‘modell-free‘ Optimierungserreichung und dem

[9] http://www.affinnova.com/
[10] Die nachfolgende Ausführungen stützen sich insbesondere auf die Darstellungen bei [SGHE07, Seite 120-129]

‘modell-driven‘ Suchprozess sind keine statistischen Analysen für die Bestimmung der optimalen Lösungen notwendig.

2.4 Technische Grundlagen

Wie aus dem Titel dieser Arbeit hervorgeht, muss die zu implementierende Anwendung eine Webapplikation sein. Hierfür werden zwei Technologien “Hibernate” und “Struts2” eingesetzt.
Worum es sich bei diesen Technologien handelt, und weshalb sie hier zum Einsatz kommen, wird nachfolgend erläutert. Es soll ein kurzer Überblick gegeben werden, ohne zu sehr ins Detail zu gehen.

2.4.1 Hibernate

Für die Entwicklung der EvoMarketOnline-Anwendung wird die objektorientierte Programmiersprache Java eingesetzt werden. Die für die Befragungen benötigten Daten werden daher als Java-Objekte im Speicher gehalten. Diese Daten müssen aber auch persistent, z. B. in einer Datenbank, gespeichert werden können.
Eine Möglichkeit besteht darin, das so genannte O/R-Mapping über eine JDBC-Verbindung selbst in die Hand zu nehmen. Hierbei wird die Kommunikation zwischen Java und der Datenbankenbank ausschließlich über SQL-Befehle durchgeführt [Sta05, Seite 277]. Daher sind Kenntnisse über die Interna der Datenbanktabellen unabdingbar. Änderungen in der Datenbank wirken sich direkt auf die SQL-Abfragen in Java aus. Es existiert demnach eine sehr starke Kopplung zwischen Programmlogik und Datenbank. Hibernate verfolgt einen anderen Ansatz. Die Datenerhaltung soll aus Sicht des Programmierers möglichst transparent sein [Sta05, Seite 271]. Das bedeutet, dass er, unabhängig von der tatsächlich eingesetzten Datenbank und den verwendeten Datenbanktabellen, programmieren können soll. Um etwa ein Java-Objekt in der Datenbank zu speichern, ist kein SQL-Befehl mehr nötig. Es reicht, dem Objekt mitzuteilen, sich persistent zu speichern. Hibernate sorgt dann für die Speicherung der Daten in der Datenbank. Um das O/R-Mapping mit Hibernate realisieren zu können, muss für jede Java-Klasse, die in der Datenbank gespeichert werden soll, eine XML-Datei angelegt werden [Sta05, Seite 272f]. Diese legt fest, wie die Verknüpfung der Java-Objekte mit den Tabellen der verwendeten Datenbank geschehen soll. Über diese XML-Datei ist Hibernate dann in der Lage, das O/R-Mapping durchzuführen. Weiterführende Informationen über die Funktionsweise von Hibernate sind unter `http://www.hibernate.org` und in der Hibernate Referenz Dokumentation unter `http://www.hibernate.org/5.html` zu finden.
Neben Hibernate existiert das von der Firma Sun entwickelte Persistenz-Framework Java Persistenz API (JPA). Informationen zum Thema JPA gibt es unter `http://java.sun.com/javaee/technologies/persistence.jsp`.

2.4.2 Struts2

Bei der EvoMarketOnline-Anwendung handelt es sich um eine Webapplikation. Sie besteht aus Java-Klassen, in denen die Programmlogik enthalten ist, und JSP-Seiten, die für die Präsentation der Daten verantwortlich sind.
In der zu entwickelnden Anwendung soll es eine saubere Trennung von Programmlogik und Präsentation nach dem MVC-Prinzip geben. Dieses sieht vor, dass das Model (Programmlogik) und die View (Präsentation auf dem Bildschirm) voneinander getrennt sind. Die Synchronisation dieser beiden Schichten wird durch eine dritte Komponente Namens Controller vorgenommen. MVC entkoppelt Model und View, um die Flexibilität und die Wiederverwendbarkeit der einzelnen Komponenten zu steigern [GHJV01].
Struts2 ist ein Open Source Framework, das von der Apache Software Foundation entwickelt und gefördert wird.
Im Rahmen dieser Arbeit wird Struts2 eingesetzt werden, um ein sauberes Design zu ermöglichen und eine Verbesserung der Wartung des Codes zu erzielen.
Auch für Struts2 gibt es ein Konkurrenzprodukt. Es wird von der Firma Sun entwickelt und nennt sich Java Server Faces (JSF). Unter `http://java.sun.com/j2ee/javaserverfaces/` können weitere Informationen zu JSF eingeholt werden.

3 Vorhandene Lösungen und Anforderungsanalyse

Wie bereits in der Einleitung erwähnt, ist diese Arbeit in Zusammenarbeit mit dem Arbeitsbereich Marketing und Innovation der Universität Hamburg entstanden.
Der Arbeitsbereich Marketing und Innovation der Universität Hamburg (AMI) setzt für die evolutionäre Marktforschung die Anwendung “Evomarket” ein, welche im Rahmen des Studienprojektes “Evolutionäre Marktforschung” entwickelt wurde und nur auf ein Einplatzsystem lauffähig ist.
Das Studienprojekt ist in Zusammenarbeit zwischen AMI und Borco, ein unabhängiger Produzent und Vermarkter von internationalen Spirituosen und Weinen, zur Designgestaltung einer Tequila-Marke entstanden.
Um die Anforderungen besser zu ermitteln, wird in diesem Kapitel zunächst einmal die Ausgangslage in Augenschein genommen. Dies hilft die einzelnen Anwendungsfälle zu ermitteln und zu analysieren. Später können so die optimierten und die weggefallenen Abläufe besser herausgestellt werden.

3.1 Ist-Analyse

Die JAVA-Anwendung “Evomarket” besitzt eine sehr einfache Oberfläche für die Durchführung der Befragungen, deren Ziel es ist, ein persönliches Optimum für den Probanden mit Hilfe von EA zu ermitteln (Abb. 3.1).
Bevor die Befragung durchgeführt werden kann, müssen die Probandendaten “Alter”, “Geschlecht” und “Beruf” eingegeben werden.
Die Befragung fängt anschließend mit einer zufällig generierten Startpopulation an.
Der Proband hat dann die Möglichkeit mittels eines einfachen oder zweifachen Klicks eine oder zwei Flaschen als Eltern für die nächste Generation auszuwählen.
Bei einem zweifachen Klick wird die selektierte Flasche sowohl als Vater als auch als Mutter übernommen.
Aus beiden Eltern werden unter Anwendung von EA genauso viele Individuen erzeugt, wie die angegebene Größe der Population.
Die Befragung endet nach einer angegebenen Anzahl der Generationen oder wird von dem Benutzer beendet.
Nach dem Ende der Befragung besteht die Möglichkeit das Resultat in eine Datenbank zu speichern.
Die in der Datenbank gespeicherten Resultate können später als “PDF” exportiert (Abb. 3.3) und mit Hilfe von der Conjoint-Analyse bewertet werden.

Abbildung 3.1: Evomarket

Wie man aus der Abbildung 3.2 entnehmen kann, bietet die Anwendung “Evomarket” folgende Einstellmöglichkeiten an:

Maximale Anzahl der Generationen:	Maximale Iterationsschritte als Abbruchkriterium.
Die Größe der Population:	Die Anzahl der Stimuli, die erzeugt und auf dem Bildschirm angezeigt werden.
Mutationsrate:	Mit Hilfe von diesem Parameter wird die Entscheidung, ob ein Gen mutiert werden soll oder nicht, getroffen. Die zulässigen Werte sind die Zahlen zwischen 0.0 und 1.0, wobei bei 1.0 jedes Gen mutiert und bei 0.0 kein Gen mutiert.
Mutationsradius:	Dieser Parameter bestimmt die Distanz der aktuellen Gen-Werte zu den anderen Gen-Werten.
Dynamische Anpassung:[1]	Mit diesem Parameter ist es möglich, den Mutationsradius dynamisch zu ändern.

Mit den Genom-Einstellungen kann man die Genome (R,G,B und Intensität) der Flaschenfarbe, die Genome der Etikettenfarbe und die Position der Etikette eingeben und das Ergebnis anzeigen lassen.[2] Die fachlichen und technischen Ziele des Projekts “Evolutionäre Marktforschung”

[1] Dieser Parameter wurde in Rahmen der Diplomarbeit von Jan Gevert mit dem Titel “Neuproduktoptimierung mit Evolutionary Conjoint für unterschiedliche Nutzenstrukturen auf individuelle Optima: Eine experimentelle Untersuchung” implementiert.

[2] Die Felder wurden nachträglich beschriftet und gefärbt

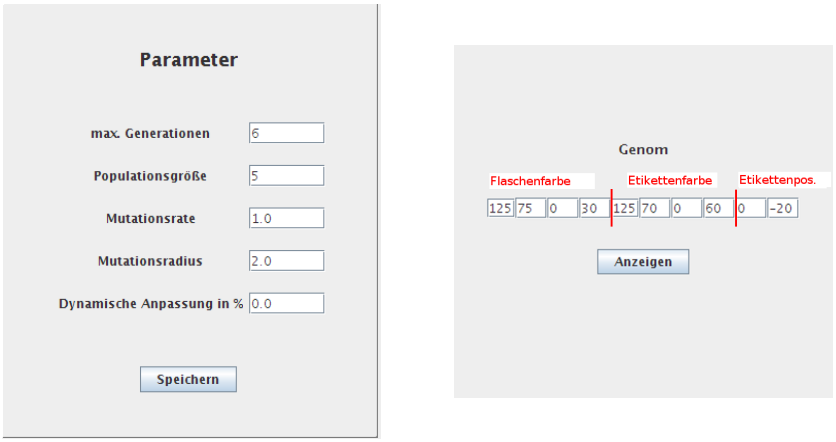


Abbildung 3.2: Einstellmöglichkeiten von Evomarket

werden in der Projekt-Dokumentation wie folgt definiert:
[3] Als fachliches Ziel wird die fundierte Erkundung von Konsumentenpräferenzen bezüglich der Designgestaltung für ein Trendgetränk positionierte imaginäre Rummarke unter Berücksichtigung folgender Eigenschaften angegeben.

- Die Flaschenfarbe
- Die Etikettenfarbe
- Die Position der Etikette

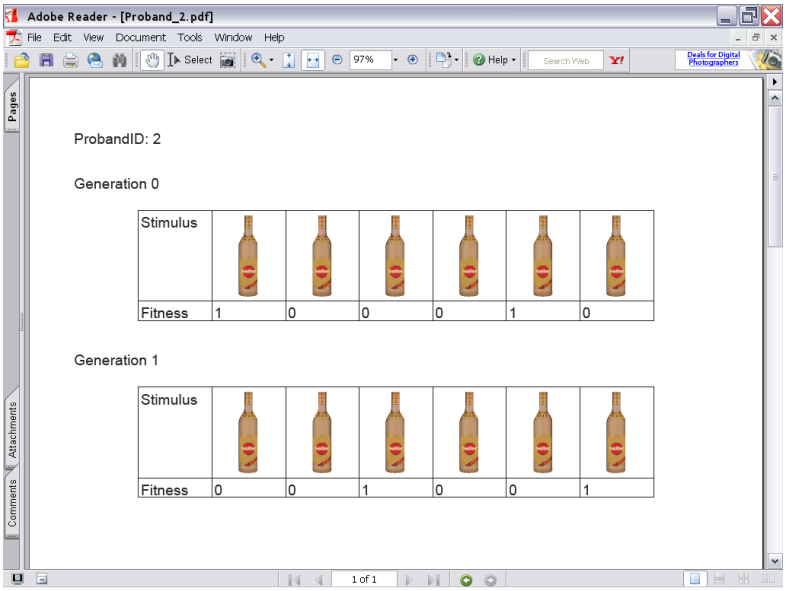


Abbildung 3.3: Evomarket PDF Export

[3] Die nachfolgende Ausführungen beziehen sich auf die Dokumentation des Projekts “Evolutionäre Marktforschung”

Folgende Punkte werden als technische Ziele der zu implementierenden Anwendung festgesetzt:

- Sie soll interaktiv sein
- Sie soll die oben genannten Eigenschaften der imaginären Flasche mit Hilfe von EA verändern
- Sie soll die Möglichkeit für die Anpassung der EA-Parameter bieten
- Sie soll die Daten persistieren, um die Analyse der Konsumentenpräferenzen zu ermöglichen

Bei der Betrachtung der festgesetzten Ziele sieht man, dass sie sehr allgemein und produktspezifisch gehalten sind.
Obwohl die Anwendung die oben genannten Ziele erreicht hat, weist sie mehrere Schwachstellen auf.
Im Folgenden werden diese Schwachstellen genauer in Betracht gezogen.

3.2 Schwachstellenanalyse

In diesem Unterkapitel werden die Schwachstellen, die die Anwendung “Evomarket” aufweist, durch das Anwendungsfall-Diagramm dargestellt und anschließend die softwaretechnischen Ursachen für diese Schwachstellen analysiert.
Die Anwendungsfälle werden folgendermaßen erläutert (Vgl. [Sch05, 217]):

- **Name:** Der Name des Anwendungsfalles.
- **Beschreibung:** Die Beschreibung des Anwendungsfalles.
- **Vorbedingungen:** Alle Bedingungen, die für die Durchführung des Anwendungsfalles erfüllt sein müssen.
- **Ablaufschritte:** Die normalen Ablaufschritte des Anwendungsfalles.
- **Nachbedingungen:** Das erwartete Ergebnis nach dem erfolgreichen Durchlauf des Anwendungsfalles.
- **Ausnahmen:** Die unerwarteten Ergebnisse nach dem erfolgreichen Durchlauf des Anwendungsfalles.

Der erste Anwendungsfall stellt die allgemeine Benutzung der Anwendung durch den Probanden dar.

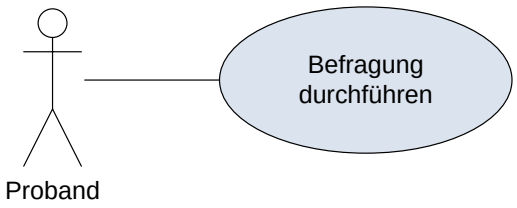


Abbildung 3.4: Befragung durchführen

Name: Starten einer Befragung (3.4)

Beschreibung:
Dieser Anwendungsfall stellt das Starten einer Befragung durch den Probanden dar.

Vorbedingungen:
Der Menüpunkt “Neuer Proband” muss betätigt werden.

Ablaufschritte:

1. Der Proband selektiert den Eintrag “Student” als Beruf, und “m” als Geschlecht.
2. Als Alter tippt er “-2147483648” ein und klickt auf den Start-Button.

Nachbedingungen:
Die Anwendung lehnt das Starten der Befragung ab.

Ausnahmen:
Die Anwendung startet die Befragung

Obwohl bei diesem Anwendungsfall die Überprüfung der Altersangabe einen Fehler produzieren sollte, startet die Befragung.
Listing 3.1 ein Auszug aus der Methode “actionPerformed(ActionEvent ev)” der Klasse “Proband-Panel” veranschaulicht den Grund dieser Schwachstelle.
Wie man aus den Zeilen 3 bis 7 entnehmen kann, beschränkt sich die Prüfung lediglich auf Typgültigkeit.
Aus der Eingabe wird eine neue Integer-Zahl erzeugt, wenn keine Ausnahmen auftereten , wird der Wert als Alter übernommen.
Somit ist es möglich, eine ganze Zahl zwischen “-2147483648” und “2147483647” als Alter einzugeben. Die Zahlen “-2147483648” und “2147483647” stellen die Unter- und Obergrenze für einen Integer-Wert dar.

Listing 3.1: Überprüfung der Altersangabe

```
1
2     if (ev.getActionCommand().equals("startEvolution")) {
3         try {this.age = new Integer(fAge.getText());
4         }
5         catch (NumberFormatException e){
```

```
6            System.err.println("Bitte Alter als Zahl eingeben");
7          }
8          this.sex = (String) fSex.getSelectedItem();
9          this.profession = (String) fProfession.getSelectedItem();
10     }
```

Die Probandendaten können bei der Auswertung wie z. B. bei der Cluster-Bildung eine wichtige Rolle spielen. Deshalb soll die Altersangabe zumindest auf ein logisches Intervall geprüft werden.

Die Hersteller haben meist bestimmte Richtlinien bezüglich des Aussehens ihrer Produkte. Deshalb wurde in der Anwendung eine Unter- und Obergrenze eingesetzt, um die Farbe des Flaschenbildes in dem vorgegebenen Farbraum zu halten.

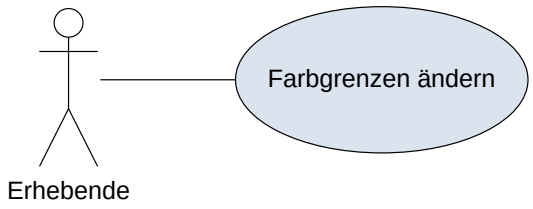


Abbildung 3.5: Ändern der Ober- und Untergrenzen der Flaschenfarbe

Name: Ändern der Ober- und Untergrenzen der Flaschenfarbe (3.5)

Beschreibung:
Dieser Anwendungsfall erläutert, diese Grenzen der Flaschenfarbe zu ändern.
Die Anwendung bietet keine Möglichkeit, diese Aufgabe mittels Oberfläche durchzuführen.

Vorbedingungen:
Der Quelltext der Anwendung muss vorliegen.
Der Erhebende muss der Programmierungssprache “JAVA” mächtig sein.
Der Erhebende kann “JAVA”-Anwendungen kompilieren.

Ablaufschritte:

1. Der Erhebende sucht die passende Stellen im Quelltext der Anwendung, trägt die gewünschten Werte ein.
2. Er kompiliert den Quelltext und startet die neu erzeugte Anwendung.

Nachbedingungen:
Die Anwendung startet.
Die Befragung kann durchgeführt werden.

Ausnahmen:
keine

Für diesen Anwendungsfall (3.5) kann zusammenfassend gesagt werden, dass eine einfache Änderung eines Produktmerkmales spezielle Kenntnisse der Programmierung erfordert. Diese Kenntnisse von einem Erhebende zu erwarten ist unzumutbar.

In der Praxis kann es häufig vorkommen, dass der Produkthersteller die phänotypischen Merkmale des Produktes verändert haben möchte.
In diesem Falle muss entweder ein Teilbild ausgetauscht oder gelöscht werden, damit eine adäquate Datenerhebung für das jeweilige Produkt überhaupt erfolgen kann.

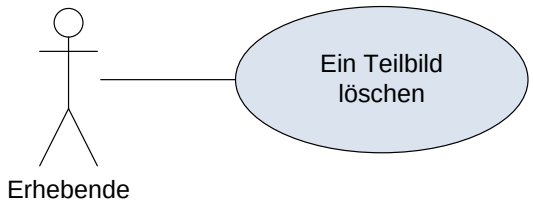


Abbildung 3.6: Löschen eines Teilbildes

Name: Löschen eines Teilbildes (3.6)

Beschreibung:
Dieser Anwendungsfall repräsentiert, wie eines der Bilder, die das gesamte Flaschenbild zusammenstellen, aus der Anwendung entfernt werden kann.

Vorbedingungen:
Wie Anwendungsfall *"Ändern der Ober- und Untergrenzen der Flaschenfarbe (3.5)"*

Ablaufschritte:

1. Der Erhebende sucht die passende Stellen im Quelltext der Anwendung, löscht die Pfadangabe des entsprechenden Bildes.
2. Er kompiliert den Quelltext und startet die neu erzeugte Anwendung.

Nachbedingungen:
Wie Anwendungsfall *"Ändern der Ober- und Untergrenzen der Flaschenfarbe (3.5)"*

Ausnahmen:
Die Befragung kann aber aufgrund der entstandenen Folgefehler nicht mehr durchgeführt werden.

Listing 3.2 verdeutlicht, weshalb das Löschen eines Teilbildes nicht so einfach funktionieren kann. Im Quelltext der Anwendung, wie die Zeilen 4, 5, 7, 10, 11, 13, 16, 17, 19, 23 und 27 zeigen, wurde fast immer davon ausgegangen, dass das Produkt fünf Teilbilder hat. Damit ist es nicht möglich, das Aussehen des Produktes zu ändern ohne mehrere Änderungen im Quelltext vorzunehmen.

Listing 3.2: Zusammenstellen der Teilbilder

```
1  bottleOp = thresholdOp(bottleR, bottleG, bottleB, bottleColorIntensity);
2  labelOp = thresholdOp(labelR, labelG, labelB, labelColorIntensity);
3  // filter bottle corpus (getImage(0))
4  int iw = getImage(0).getWidth();
5  int ih = getImage(0).getHeight();
6  BufferedImage bottle = new BufferedImage(iw, ih, BufferedImage.TYPE_INT_ARGB)
       ;
7  bottleOp.filter(getImage(0), bottle);
8
9  // filter label back (getImage(1))
10 iw = getImage(1).getWidth();
11 ih = getImage(1).getHeight();
12 BufferedImage labelBack = new BufferedImage(iw, ih, BufferedImage.
       TYPE_INT_ARGB);
13 labelOp.filter(getImage(1), labelBack);
14
15 // filter upper label (getImage(2))
16 iw = getImage(2).getWidth();
17 ih = getImage(2).getHeight();
18 BufferedImage labelUpper = new BufferedImage(iw, ih, BufferedImage.
       TYPE_INT_ARGB);
19 labelOp.filter(getImage(2), labelUpper);
20
21 // merge images
22 // a) not positioned
23 bottle.createGraphics().drawImage(getImage(3), 0, 0, null);// cap
24 bottle.createGraphics().drawImage(labelUpper, 0, 0, null);// upper label
25 // b) positioned
26 bottle.createGraphics().drawImage(labelBack, labelPosX, labelPosY, null); //
       back
27 bottle.createGraphics().drawImage(getImage(4), labelPosX, labelPosY, null);//
        front
```

Die aufgeführten Anwendungsfälle, sowie die Listings zeigen auf, dass diese Anwendung nur für dieses konkrete Produkt mit den vorgegebenen Parametern oder analog zu diesem Produkt, dieselbe Produktstruktur aufweist, wie z. B. ein Produkt mit fünf Teilbildern, ohne großen Aufwand funktionieren kann.

Als Resultat der Ist- und Schwachstellenanalyse kann festgehalten werden, dass die Anwendung die festgesetzten Ziele erreicht.
In der Praxis ist die Anwendung jedoch nur begrenzt einsetzbar, sie bietet praktisch kaum Möglichkeiten andere Produkte einzusetzen. Darüber hinaus, ist eine effektive Befragung nur dann durchzuführen, wenn die Probanden entweder die Anwendung auf eigenen Rechnern installieren oder bereit sind, sich an den Ort zu begeben, an dem die schon bereits konfigurierten Rechner zur Verfügung gestellt werden.
Beide Möglichkeiten sind nicht anwenderfreundlich, da für eine Befragung, seitens der Proban-

den, minimaler Aufwand betrieben werden sollte damit überhaupt Teilnehmende gefunden werden können. Befragungen und Umfragen sind nur verlockend, wenn keine hohen Erwartungen an den Probanden bestehen.

3.3 Anforderungsanalyse

In Kapitel 1 wurden die Anforderungen an die Online-Befragung-Anwendung bereits aufgezählt. An dieser Stelle sollen die einzelnen Punkte aufgegriffen und näher erläutert werden. Die Anforderungen lassen sich in fachliche und technische Anforderungen unterteilen.
Die Anforderungen werden erst durch ein Anwendungsfall-Diagramm dargestellt, anschließend werden die dargestellten Anwendungsfälle mit Hilfe in der Schwachstellenanalyse bereits angewandten Darstellungsform im Einzelnen erläutert.

3.3.1 Fachliche Anforderungen

Aus fachlicher Sicht gibt es zwei Benutzertypen, für die Anforderungen definiert werden müssen. Zum einen die Personen, die an der Befragung teilnehmen, also die Probanden, zum anderen die Erhebenden, die eine Befragung vorbereiten und durchführen.
Im Folgenden wird von der Annahme ausgegangen, dass ein Taschentuchhersteller die Verpackung seines Produktes nach den Wünschen der Kunden ändern möchte. Dafür beauftragt er den Arbeitsbereich Marketing und Innovation der Universität Hamburg (AMI). AMI möchte für diese Befragung die EvoMarketOnline-Software einsetzen.

3.3.1.1 Anforderungen der Probanden

Die Anforderungskriterien an den Probanden sind minimal, um genauer zu sagen es wird nur verlangt, die Befragung ohne großen Aufwand durchzuführen.
Der erste Anwendungsfall, der fachlichen Anforderungen, stellt den typischen Ablauf einer Befragung dar.

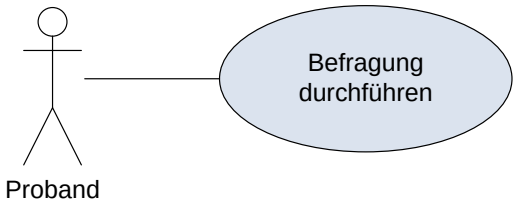


Abbildung 3.7: Befragung durchführen

Name: Befragung durchführen (3.7)

Beschreibung:
Dieser Anwendungsfall erläutert die Durchführung der Befragung durch den Probanden.

Vorbedingungen:
Der Proband muss die Adresse (URL) der Befragung aufrufen.

Ablaufschritte:

1. Die Anwendung überprüft die Zugriffrechte des Probanden (ggf. erscheint die Passwort-Abfrage).
2. Die Eingabemaske für die Probandendaten wird angezeigt.
3. Der Probanden selektiert den Eintrag “Student” als Beruf, und “m” als Geschlecht.
4. Als Alter tippt er “-2147483648” ein und klickt auf den Start-Button.
   die Anwendung lehnt die Altersangabe ab.
5. Als Alter tippt er “20” ein und klickt auf den Start-Button.

Nachbedingungen:
Die Befragung startet.

Ausnahmen:
keine

Optimale Anwendungskriterien, die der Proband an die Anwendung insgesamt gestellt, müssen minimal sein. Er erwartet eine einfache, unkomplizierte sowie nicht zeitintensive Durchführung der Befragung.

Die eigentlichen Anforderungen werden von den Erhebenden selbst gestellt.
Die nächsten Anwendungsfälle veranschaulichen die Anforderungen der Erhebenden.

3.3.1.2 Anforderungen der Erhebenden

Dass sich die Online-Befragung-Anwendung universell, zumindest bei bestimmten Produkttypen, einsetzen lässt, ist eine der wichtigsten Anforderungen der Erhebenden. Darüber hinaus soll die Online-Befragung-Anwendung folgende Möglichkeiten anbieten:

- Verschiedene EA-Parameter-Konfigurationen für eine Befragung erzeugen
- Die Befragungen absichern
- Die Ergebnisse der Befragungen ansehen und diese bewerten

Abbildung 3.8 zeigt ein Anwendungsfall-Diagramm mit den übrigen Anforderungen der Erhebenden, die die Online-Befragung-Anwendung bedienen sollen.
Nachfolgend werden diese Anwendungsfälle im Einzelnen erläutert.

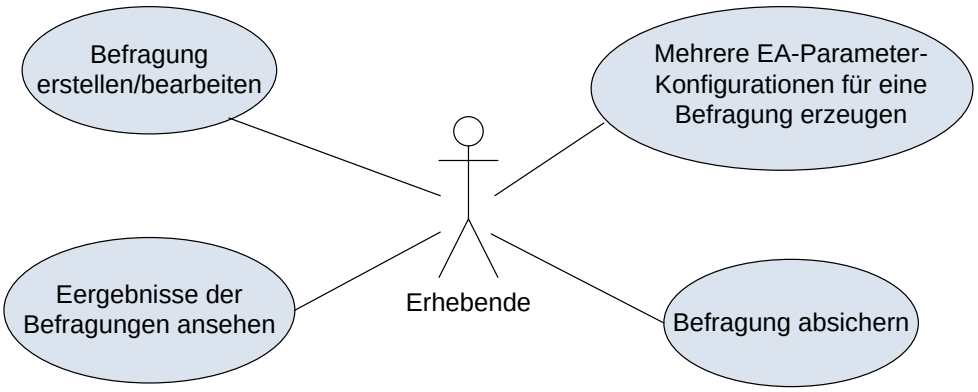


Abbildung 3.8: Anforderungen der Erhebenden

Name: Eine neue Befragung erstellen (3.8-a)

Beschreibung:
Dieser Anwendungsfall veranschaulicht die Erzeugung einer neuen Befragung mit einem neuen Produkt.

Vorbedingungen:
Die Bilder des neuen Produktes müssen zur Verfügung stehen.
Der Erhebende muss sich als Administrator anmelden

Ablaufschritte:
1. Der Erhebende klickt auf “befragung erzeugen/ändern” Link.
2. Er füllt die Felder “Name” und “Beschreibung“ aus.
3. Er weist eine neue oder vorhanden EA-Parameter-Konfiguration und eine Benutzergruppe der Befragung zu und füllt die entsprechenden Felder aus.
4. Er erstellt neues Produkt (mit Teilprodukten) , lädt dessen Bilder hoch.
5. Anschließend speichert er die Befragung ab.

Nachbedingungen:
Das neue Produkt ist in der Datenbank abgespeichert.
Alle möglichen Ausgangslinks für die Befragung werden gelistet.

Ausnahmen: keine

Name: Mehrere EA-Parameter-Konfigurationen für eine Befragung erzeugen (3.8-b)

Beschreibung:
Dieser Anwendungsfall beschreibt wie verschiedene EA-Parameter-Konfigurationen erzeugt werden können.

Vorbedingungen:
Der Erhebende muss sich als Administrator anmelden

Ablaufschritte:
1. Wählt eine Befragung aus und klickt auf “Bearbeiten” Schaltfläche an oder erstellt eine neue Befragung.
2. Auf der aufgebauten Seite kann er jetzt beliebig viele neue oder vorhandene EA-Parameter-Konfigurationen zu der aktuellen Befragung zuweisen.

Nachbedingungen:
Die Konfigurationen sind in der Datenbank abgespeichert.
Alle möglichen Ausgangslinks für die Befragung werden gelistet.

Ausnahmen: keine

Name: Eine Befragung absichern (3.8-c)

Beschreibung:
Die Befragungen müssen aus marketingtechnischen Gründen oftmals geheim gehalten werden. Deshalb soll gewährleistet werden, dass außer den dafür autorisierten Probanden niemand anderes daran teilnehmen kann und darf.
Dieser Anwendungsfall erläutert das Absichern einer Befragung.

Vorbedingungen:
Der Erhebende muss sich als Administrator anmelden

Ablaufschritte:
1. Der Erhebende wählt eine Befragung aus und klickt auf “Bearbeiten” Schaltfläche an oder erstellt eine neue Befragung.
2. Dann wählt er eine Gruppe aus, falls diese vorhanden ist, sonst legt er eine neue Gruppe an und gibt das Passwort für diese Befragung ein.

Nachbedingungen:
Die Befragung ist in der Datenbank abgespeichert und darf nur durch erfolgreiche Passwort-Eingabe durchgeführt werden.

Ausnahmen: keine

Name: Ansehen der Befragungsergebnisse (3.8-d)

Beschreibung:
Der Erhebende möchte die Ergebnisse einer Befragung darstellen und bewerten

Vorbedingungen:
Die Befragung muss durchgeführt worden sein
Der Erhebende muss sich als Administrator anmelden

Ablaufschritte:
1. Der Erhebende listet alle Befragungen auf, sucht die entsprechenden aus, klickt diese an.
2. Dann werden die Ergebnisse als Tabellenform dargestellt.
3. Die Ergebnisse können nun mit geeigneten Statistikprogrammen, wie z. B. SPSS, R, SAS, AMOS usw., evaluiert werden.

Nachbedingungen:
Die Ergebnisse werden dargestellt.

Ausnahmen: keine

Neben diesen fachlichen Anforderungen werden noch technische Anforderungen an die Online-Befragung-Anwendung gestellt. Als nächstes werden diese Anforderungen beschrieben.

3.3.2 Technische Anforderungen

3.3.2.1 Online Verfügbarkeit

Wie bereits aus dem Titel dieser Diplomarbeit hervorgeht, muss EvoMarketOnline online verfügbar gemacht werden.
Hierzu kämen ein Dutzend Technologien hinzu, die diese Anforderung erfüllen können.
Die Anwendung soll ohne zusätzliches Zutun der Probanden laufen und eine einfache, unkomplizierte und gewohnte Bedienung bieten. Des Weiteren sollen die Kernkomponenten der vorhandenen Lösung benutzt werden, um den Prototyp der EvoMarketOnline-Anwendung in der Bearbeitungszeit dieser Diplomarbeit zu entwickeln. Darüber hinaus sollen keine Mehrkosten aufgrund bei der Entwicklung eingesetzter Technologien für den Arbeitsbereich Marketing und Innovation der Universität Hamburg entstehen.

3.3.2.2 Administrierbarkeit über ein Web-Interface

Die zu entwickelnde Anwendung muss administriert werden. So ist etwa das Erzeugen und das Bearbeiten der Befragungen notwendig. Diese Administrationsaufgaben sollen über ein Web-Interface möglich sein.

3.3.2.3 Persistenz der Daten

Die erzeugten Befragungen und deren Ergebnisse müssen für spätere Benutzung bzw. Auswertungen dauerhaft gespeichert werden.

Nachdem die vorhandene Lösung ”Evomarket“ und die Anforderungen an die in Rahmen dieser Diplomarbeit zu implementierende EvoMarketOnline-Anwendung vorgestellt wurden, wird in dem nächsten Unterkapitel versucht, den Soll-Zustand der Lösung zu analysieren.

3.4 Soll-Analyse

Die von dem Arbeitsbereich Marketing und Innovation der Universität Hamburg gestellten Anforderungen entsprechen den Grundanforderungen für eine EA basierte Marktforschung-Anwendung. Eine ideale Software-Lösung für evolutionäre Marktforschung soll mehr als die oben genannten Anforderungen erfüllen. Sie soll:

- Algorithmus-Auswahl für die Befragungen anbieten.
- alternative Bilder für ein Teil des Produktes bearbeiten.
- die Ergebnisse der Befragungen statistisch darstellen und gegebenenfalls diese für die gängigen Statistikprogramme wie z. B. SPSS, R, SAS und AMOS exportieren.

Als nächstes werden die oben genannten Anforderungen näher erklärt.

Algorithmus-Auswahl für die Befragungen

Die Ist-Lösung ”Evomarket“ setzt als Algorithmus die unter 2.2.2 beschriebene ”μ, λ“-ES oder $\mu/\rho, \lambda$-ES ein, je nachdem, wie viele Individuen ein Proband ausgewählt hat, wobei$\mu = \lambda$ ist.
Bei der Selektion eines Individuums wird ”μ, λ“-ES, bei der Selektion von zwei Individuen ”$\mu/\rho, \lambda$“-ES, angewandt. Beide Varianten unterscheiden sich bei der Erzeugung der Nachkommen bei μ, λ wird ein Elter einfach dupliziert und bei $\mu/\rho, \lambda$ werden beide Eltern $\rho = 2$ rekombiniert (s. o.).
Weil diese Strategie relativ gute Ergebnisse für Befragungen liefert, bei denen die Farbe und die Position der Komponente optimiert werden sollen, wird diese auch bei der Entwicklung der EvoMarketOnline-Anwendung eingesetzt.
Weil die Qualität des Ergebnisses sowohl von der Art der Befragung ,als auch von der eingesetzten ES abhängig ist, soll die ideale Lösung die Möglichkeit anbieten, unterschiedliche EA Varianten wie etwa $\mu + \lambda$ oder $\mu/\rho + \lambda$ für einzelne Befragungen auszuwählen.

Alternative Bilder für ein Teil des Produktes bearbeiten

Um diese Anforderung näher zu beschreiben, wird wie auch in den vorherigen Anwendungsbeispielen von der Annahme ausgegangen, dass ein Taschentuchhersteller das Produktdesign optimieren möchte. Neben den Produkteigenschaften wie der Positionierung der Etikette, der Farbe, soll auch die Präferenz des Kunden hinsichtlich der beiden vorgegebenen Etikettendesigns ermittelt werden. Die vorliegende Lösung und EvoMarketOnline können keine Ermittlung über die Präferenz der beiden vorgegebenen Etikettendesigns machen. Es gibt keine Möglichkeit herauszufinden, welche Gewichtung diese Komponente bei der Entscheidungsfindung des Probanden hat. Deshalb ist es notwendig, eine Vorab-Befragung durchzuführen, um eine erste Ein - und Abschätzung über die Präferenz des Probanden hinsichtlich dieser Teilkomponente machen zu können.
Darüberhinaus muss die Software in der Lage sein, die Präferenzen des Probanden in EA-Parameter umzuwandeln und für jeden Probanden eine personalisierte Konfiguration erzeugen, damit die eigentliche Befragung, mit Hilfe der Konfiguration durchgeführt werden kann.

Statistische Darstellung und Export der Ergebnisse

Die Ergebnisse der Befragungen werden bei ”Evomarket” für jeden Probanden in Form von “PDF”-Dateien ausgedruckt, die Online-Befragung-Anwendung hingegen stellt die Ergebnisse als Tabelle für eine Befragung dar.
Diese beiden Darstellungsmethoden geben zwar eine Übersicht der Ergebnisse, reichen aber nicht für deren fachgerechte Bewertungen aus.
Die Ergebnisse sollten graphisch dargestellt werden. Die Visualisierung soll Auskunft über gängige statistische Parameter wie z.B. Median, Modus, arithmetisches Mittel usw. geben. Werden detailliertere Ergebnisse bzw. Darstellungsformen erwünscht, sollen diese in geeignete Dateiformate für gängige Statistikanwendungen exportiert werden.

In diesem Kapitel wurden die vorhandene Lösung “Evomarket” und deren Schwachstellen sowie die Anforderungen des Arbeitsbereiches Marketing und Innovation der Universität Hamburg an “EvoMarketOnline“ vorgestellt. Anschließend wurden die Anforderungen an eine “ideale Lösung” betrachtet. Sie sind realisierbar, jedoch würde die Umsetzung den inhaltlichen sowie zeitlichen Rahmen der Bearbeitung dieser vorliegenden Diplomarbeit sprengen.

4 Systementwurf und Realisierung

In diesem Kapitel wird auf den Entwurf der EvoMarketOnline-Anwendung sowie die damit verbundene Implementierung eingegangen.

4.1 Systementwurf

Bevor einzelne Komponenten der EvoMarketOnline-Anwendung im Detail besprochen werden, soll hier ein kurzer Gesamtüberblick über die Architektur gegeben werden.

4.1.1 Architekturüberblick

Die unten abgebildete Architektur der EvoMarketOnline-Anwendung (siehe Abb. 4.1) ist in drei Schichten aufgeteilt:

- die Persistenzschicht
- die Anwendungsschicht
- die Präsentationsschicht

somit spiegelt diese Architektur die klassische Drei-Schichten-Architektur wieder.
Laut ([DH07]) bilden die Schichten der klassischen Drei-Schichten-Architektur die grundsätzlichen Aufgaben von Softwaresystemen ab und haben sich in der Softwareentwicklung bewährt.
Die Aufgaben und Verantwortlichkeiten der Schichten sind voneinander abgegrenzt und lose aneinander gekoppelt. Diese lose Kopplung ermöglicht die Wiederverwendbarkeit von Software-Komponenten und die Änderbarkeit der Schichten unter der Voraussetzung, dass die Zugriffschnittstellen zwischen den Schichten nicht verändert werden.

Persistenzschicht

Diese Schicht bildet die unterste Schicht der Architektur und ist für die dauerhafte Speicherung der notwendigen Daten der Anwendung zuständig. Für die Speicherung wird eine Relationale Datenbank eingesetzt, dabei erfolgt der Zugriff aus der Anwendungsschicht nicht direkt sondern mit Hilfe von einer Zugriff-Schnittstelle.

Anwendungsschicht

Die Anwendungsschicht kapselt die eigentliche Anwendungslogik und befindet sich eine Ebene über der Persistenzschicht. Die “Evolution“ ist die Kernkomponente dieser Schicht.
Sie bildet ein Individuum bzw. das ”Produkt“ ab und implementiert den konkreten Evolutionären Algorithmus für die EvoMarketOnline-Anwendung. Die Modell-Komponente verwaltet die Kernkomponente ”Evolution“ und beinhaltet zusätzlich die Modell-Objekte des objektorientierten Datenmodells.

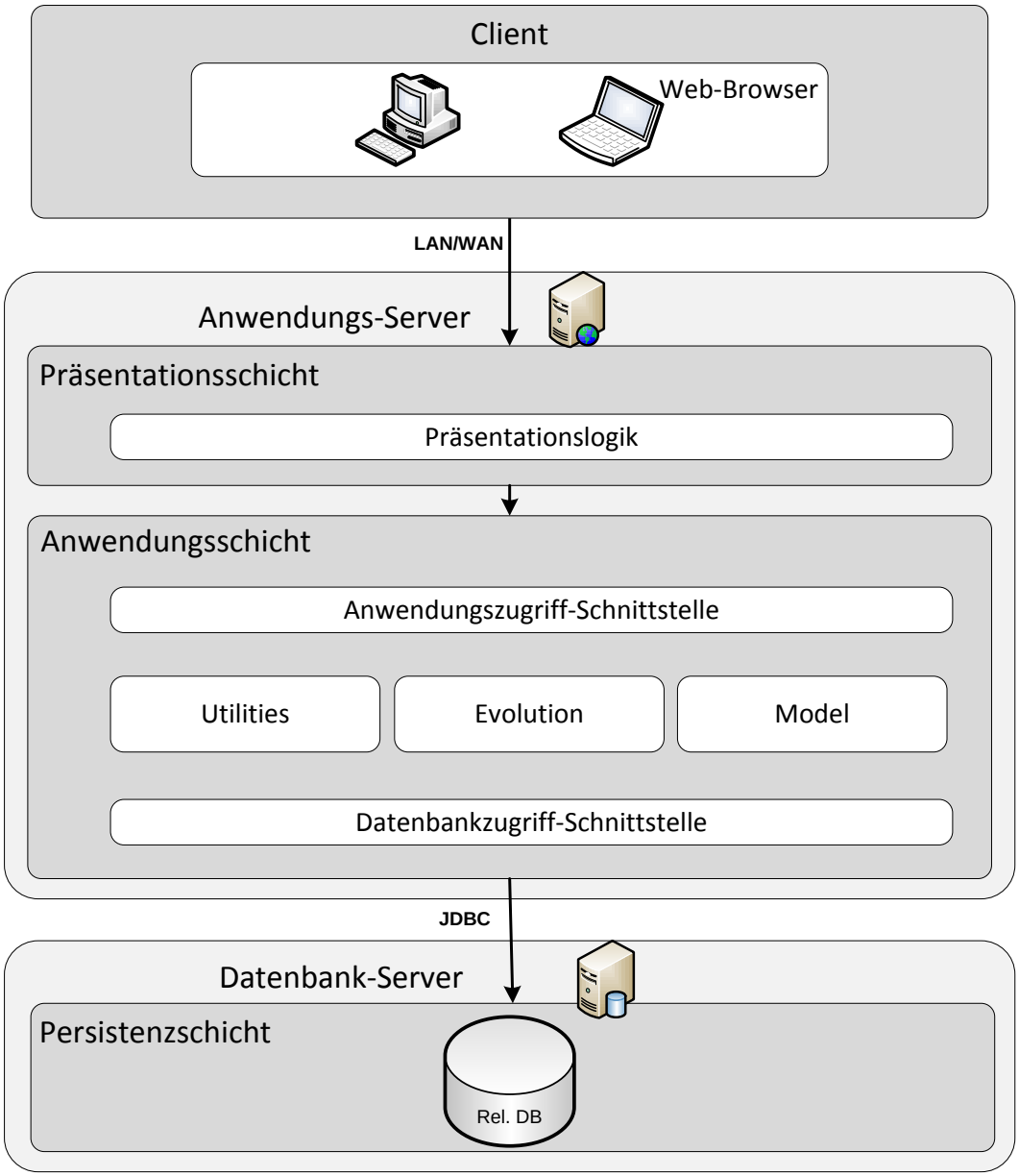


Abbildung 4.1: Gesamtüberblick über die Architektur der EvoMarketOnline-Anwendung

Die Utilities-Komponente, beinhaltet Hilfsfunktionen, die von den anderen Komponenten genutzt werden. Beispielsweise stellt sie Funktionen zum Zusammensetzen und oder zum Erzeugen der Produktbilder zur Verfügung. Die ”Anwendungszugriff-Schnittstelle“ dient als Fassade (vgl. [GHJV01])für die Anwendungsschicht. Alle Zugriffe aus der Präsentationsschicht auf die Anwendungsschicht erfolgen über diese Komponente.

Präsentationsschicht

Die Präsentationsschicht repräsentiert die Anwendungsschicht nach außen hin und ermöglicht die Interaktion mit dem Benutzer. Weil es sich um eine Web-Applikation handelt, erfolgt der Zugriff über einen Web-Browser.
Für die Darstellung der gewünschten Webseiten ist die Komponente ”Präsentationslogik“ zuständig.

4.1.2 Architektur im Detail

Im vorangegangenen Abschnitt wurde die EvoMarketOnline-Anwendung in einer abstrakten Architekturübersicht vorgestellt.
In diesem und dem nächsten Abschnitt folgen nun detailliertere Erläuterungen zu einzelnen Komponenten innerhalb der Schichten, die aus Architektursicht interessant sind.
Weitere implementierungsspezifische Details werden in Unterkapitel 4.2.2 erläutert.

4.1.2.1 Persistenzschicht

Es ist zu erwarten, dass in der Anwendung große Datenmengen gespeichert werden. Die Anzahl der Produkte und deren Teilprodukte werden, abhängig von den durchgeführten Befragungen und von der Teilnehmerzahl, mit der Laufzeit der Anwendung stark wachsen.
Außerdem sind Anfragen verschiedenster Art auf die Daten zu erwarten. Typische Anfragen sind z. B. *”Gib mir alle Ergebnisse für die Befragung ’T-Shirt Optimierung’“* oder *”Zeige alle Basis-Individuen für die Befragungen“*.
Damit diese großen Datenmengen bewältigt werden können, sollen die Daten auf einer Datenbank gespeichert werden.
Im konkreten Fall wurde für die persistente Speicherung der Daten eine relationale Datenbank mit dem folgenden Datenbankschema (siehe Abb. 4.2) eingesetzt.

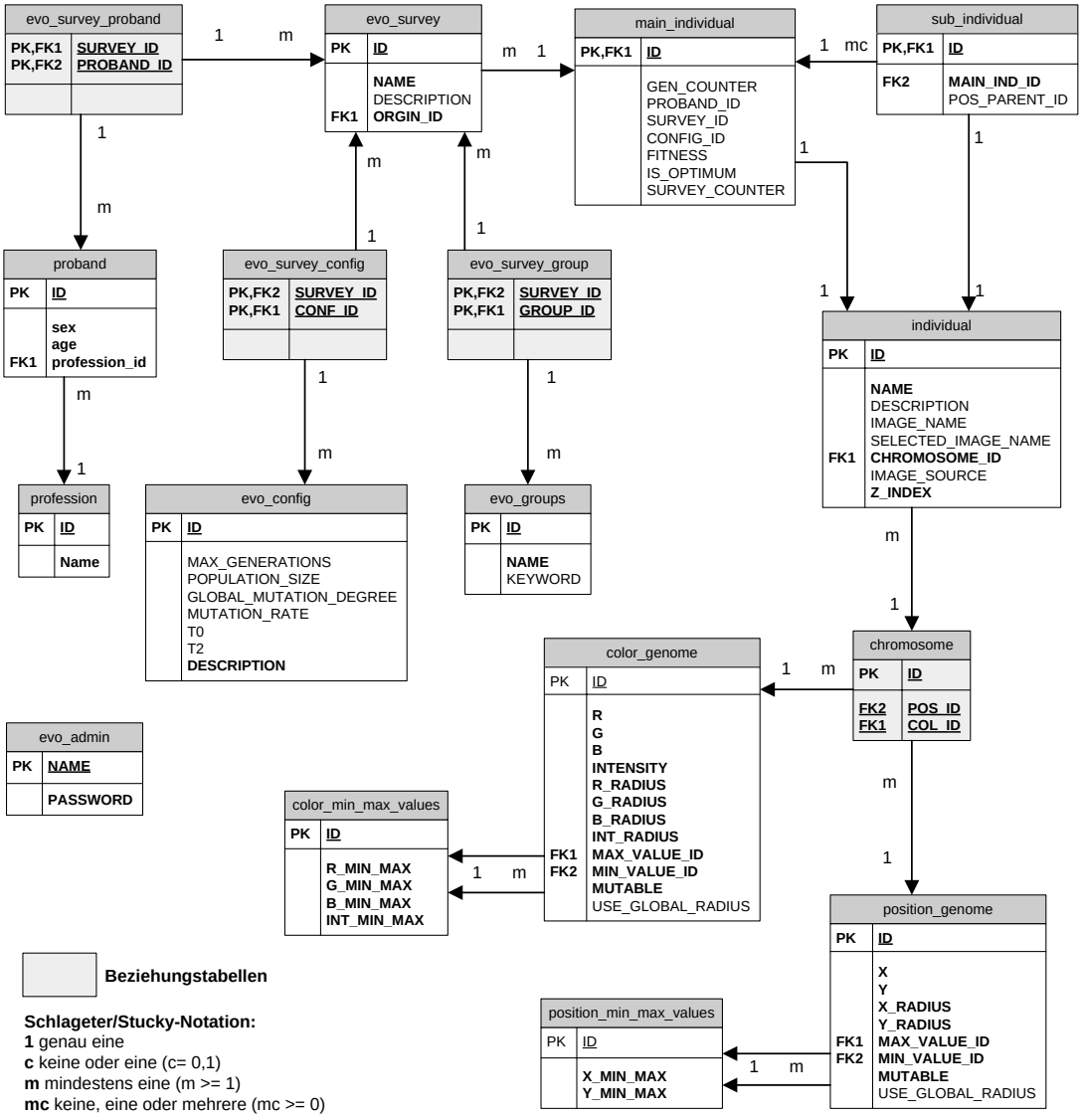


Abbildung 4.2: Tabellen und Relationen des Datenbankschemas

Im Folgenden werden die einzelnen Entitäten näher beschrieben.

evo_survey

Die *evo_survey*-Entität ist die zentrale Tabelle, in der vorbereitete Befragungen gespeichert werden. Neben einer ID (*id*) hat jede *Befragung* einen Namen *(NAME)*, eine Beschreibung *(DESCRIPTION)* und ein Basis-Individuum *(ORGIN_ID)*.

individual, main_individual, sub_individual

Diese drei Tabellen zusammen bilden das ”Produkt-Objekt“ aus der Anwendungsschicht auf der Datenbankebene. Ein Produkt-Objekt besteht aus einem ”Main-Individual“-Objekt und aus einem oder mehreren ”Sub-Individual“-Objekte, wobei diese Objekte wiederum ”Individual“-Objekte sind.
Im Folgenden werden die ”Main-Individual“-Objekte als *”Basis-Individuum*, die ”Sub-Individual“-Objekte als *”Teil-Individuum“*, die ”Individual“-Objekte als *”Individuum“* und das ”Produkt-Objekt“ als *”Produkt“* genannt.
Die Gemeinsamkeiten aller Individuen werden in der Tabelle ”individual“ gespeichert. Jedes Individuum hat eine *(ID)*, einen Namen *(NAME)* und eine Beschreibung *(DESCRIPTION)*. Diese drei Attribute stellen allgemeine Eigenschaften eines Individuums dar.
”IMAGE_NAME“, ”SELECTED_IMAGE_NAME“, ”IMAGE_SOURCE” und *“Z_INDEX”* speichern die Bild-Informationen eines Individuums ab. Dabei werden die Bilder in der “IMAGE_SOURCE”-Spalte als “BLOB” gespeichert, dies tritt nur bei den Individuen ein, die zusammen das Basis-Individuum einer Befragung darstellen.
In der ”main_individual“-Entität werden die Beziehungen zwischen dem Individuum und der Befragung, der Konfiguration und dem Probanden mit Hilfe von *”SURVEY_ID“*, *”CONFIG_ID“* und *”PROBAND_ID“* hergestellt.
Bei Ablauf einer Befragung werden mehrere Produkt-Generationen erzeugt. Um die Evolution nachzuvollziehen müssen diese Informationen festgehalten werden. Die Spalten *”GEN_COUNTER“* und *”FITNESS“* speichern die Daten, zu welcher Generation ein Individuum gehört und ob es von dem Probanden für die nächste Generation ausgewählt wurde. Ein Proband kann eine Befragung mehrere Male durchführen, um das später zu bestimmen, werden entsprechende Informationen in der *”SURVEY_COUNTER“*-Spalte gespeichert. *”IS_OPTIMUM“* bestimmt ob ein Individuum von einem Probanden als persönliches Optimum ausgewählt wurde.

Die spezielle Daten der Teil-Individuen werden in der Tabelle ”sub_individual“ abgelegt. Jedes Teil-Individuum hat ein Basis-Individuum als Elter, kann aber gleichzeitig an einem Teil-Individuum angebunden werden. Diese Auskünfte werden unter *”MAIN_IND_ID“* und
”POS_PARENT_ID“ festgehalten.

Die Primärschlüssel *”ID“* von beiden Tabellen ”main_individual“ und ”sub_individual“ sind gleichzeitig Fremdschlüssel, die auf die Tabelle ”Individual“ verweisen. Diese drei Tabellen könnten auch als eine einzige Tabelle dargestellt werden aber aus Performanzgründen wurden sie in drei geteilt.

chromosme

Jedes Individuum hat genau ein Chromosom *“CHROMOSOME_ID”* (siehe Tabelle ”individual“), ein Chromosom hat genau ein Farb-Gen *“COL_ID”* und ein Position-Gen *“POS_ID”*. Diese Tabelle verbindet die Tabelle ”individual“ mit den Tabellen ”color_genome“ ”und position_genome”.

color_genome

In dieser Entität werden die Farbinformationen wie *“R”*, *“G”*, *“B”* und Intensität *“INTENSITY”* eines Individuum bzw. dessen Bild gespeichert.
In EvoMarketOnline findet die Mutation eines Bildes prozentual statt. Deshalb sind diese Werte die Änderungen gegenüber den ursprünglichen Farbwerten. Hierfür sind die Werte zwischen -100 (%) und 100 (%) zulässig.
Darüber hinaus wird der spezielle EA-Parameter “Mutation-Radius” für jede Farb-Eigenschaft *“R_RADIUS”*, *“G_RADIUS”*, *“B_RADIUS”* und *“INT_RADIUS”* auch in dieser Tabelle festgehalten. Die Benutzung dieser Werte im Algorithmus ist von dem Wert von *“USE_GLOBAL_RADIUS”* abhängig. Das Attribut *“MUTABLE”* bestimmt, ob die Farbwerte überhaupt verändert werden dürfen oder nicht.
Um die Farbwerte in vorgegeben Grenzen zu ändern, hat jedes Farb-Gen eine Ober- und Untergrenze. Diese werden in der Tabelle “color_min_max_values“ gespeichert und werden durch Fremdschlüssel *“MAX_VALUE_ID”* und *“MIN_VALUE_ID”* referenziert.

color_min_max_values

Die Änderungen der Farbwerte eines Individuums müssen sich produktspezifisch in einem zulässigen Intervall bewegen.
In dieser Tabelle werden Grenzwerte für jeden der Farbwerten R *(“R_MIN_MAX”)*, G *(“G_MIN_MAX”)*, B *(“B_MIN_MAX“)* und Intensität *(”INT_MIN_MAX“)* gespeichert.

position_genome

Die implementierte Anwendung ist in der Lage den EA-Algorithmus auch auf die Position eines Teil-Individuums bzw. dessen Bildes anzuwenden. Diese Start-Koordinaten des Bildes werden in der Spalten , *“X”* und *“Y”* gespeichert, wobei diese relativ zu übergeordneten Bild-Koordinaten sind. Hier sind elf stellige negative und positive Integer-Werte erlaubt. Die anderen Attribute sind Analog zu “color_genome”-Tabelle

position_min_max_values

Die Änderungen der Farbwerten eines Individuums müssen sich produktspezifisch in einem zulässigen Intervall bewegen.
In dieser Tabelle werden Grenzwerte für jeden der Farbwerten R *(“R_MIN_MAX”)*, G *(“G_MIN_MAX”)*, B *(“B_MIN_MAX“)* und Intensität *(”INT_MIN_MAX“)* gespeichert. Diese Tabelle ist Analog zu “color_min_max_values” zu verstehen. In dieser Tabelle werden minimale und maximale Werte für die Position-Gen gespeichert.

evo_config

Die von dem EA-Algorithmus benötigten EA-Parameter werden in dieser Tabelle gespeichert.

Neben einer Id *(ID)* und eine Beschreibung *(DESCRIPTION)* hat die Konfiguration folgende EA-Parameter:

- "MAX_GENERATIONS" bestimmt die Anzahl der Evolution-Schritte.
- "POPULATION_SIZE" ist die Zahl der zu repräsentierenden Individuen auf dem Bildschirm.
- "GLOBAL_MUTATION_DEGREE" stellt den Abstand der neu berechneten Gen-Werte zu den alten Gen-Werten dar.
- "MUTATION_RATE" bestimmt, ob ein Gen mutiert werden soll oder nicht.
- "T0" und "T2" sind konstante Strategieparameter, die in einem ES-Algorithmus eingesetzt werden (siehe Seite 14).

evo_groups

Einige Befragungen müssen für bestimmte Gruppen von Probanden sichtbar sein. Dies wird mit Hilfe von in der "evo_groups"-Entität gespeicherten Daten umgesetzt. Eine Probanden-Gruppe hat eine Id *("ID")*, einen Namen *("NAME")* und ein Schlüsselwort *("KEYWORD")*. Das Schlüsselwort fungiert als Passwort für eine Befragung, ist aber optional, wenn das gesetzt ist, muss dieses von den Probanden beim Beginn der Befragung eingegeben werden.

proband

In dieser Tabelle werden die Daten der Probanden einer Befragung wie das Geschlecht *("sex")*, das Alter *("age")* und die Tätigkeit-Id *("profession_id")* gespeichert. Die Tätigkeit wird in der Entität "profession", die nur aus einer *("ID")* und einem Namen *("NAME")* besteht, festgehalten.

evo_admin

Die Befragungen dürfen nur von den Benutzern, die in dieser Tabelle gespeichert sind, erzeugt, bearbeitet und angesehen werden. Ein Administrator hat einen Namen *("NAME")* und ein Passwort *("PASSWORD")*, das er bei den oben genannten Operationen eingeben muss.
Der Namen ist gleichzeitig Primärschlüssel dieser Tabelle.

Beziehungstabellen: evo_survey_config, evo_survey_groups, evo_survey_proband

Eine Befragung hat neben den oben genannten Attributen mehrere Konfigurationen, mehrere Gruppen und mehrere Probanden. Diese Tabellen verknüpfen eine Befragung mit den Konfigurationen, mit den Gruppen und mit den Probanden.

4.1.2.2 Datenbankzugriff-Schnittstelle

Um die Persistenzschicht von der Anwendungsschicht sauber zu trennen, wurde die Datenbankzugriff-Schnittstelle implementiert. Jeder Zugriff aus der Anwendungsschicht auf den Datenbank-Server erfolgt über diese Schnittstelle. Damit kann die Anwendungsschicht unabhängig von dem Datenbank-Server entwickelt werden, was auch eine einfache Erweiterbarkeit, Wiederverwendbarkeit und Wartbarkeit der Anwendung bzw. einzelner Komponenten zur Folge hat. Mögliche Änderungen der Datenbank wirken sich nicht unmittelbar auf die Anwendungsschicht aus.

Die Anwendungsschicht beruht auf ein objektorientiertes Datenmodell, weshalb in der Datenbankzugriff-Schnittstelle auch gleichzeitig ein Objekt-Relationales Mapping umgesetzt wird. Dabei werden die Tabellen der Datenbank in ein objektorientiertes Modell übertragen. Die notwendigen Objekte sind Teil der Komponenten “Modell” und “Evolution”. Sie werden im nächsten Unterkapitel behandelt.

4.1.2.3 Anwendungsschicht

Die Evolution-Komponente und Modell-Komponente stellen die Abbildung des Datenbankschemas als Objekte eines objektorientierten Datenmodells dar. Die Daten aus der Datenbank werden zur Verarbeitung in diese Objekte geladen. Nach der Verarbeitung werden sie zur persistenten Speicherung wieder in die Datenbank geschrieben.
In UML-Klassendiagrammen [1] in Abbildung 4.3 und Abbildung 4.4 ist zu sehen, dass das Objektmodell dem Datenmodell (siehe Abbildung 4.2) stark ähnelt. Lediglich die Beziehungen ohne zusätzliche Attribute (evo_survey_config, evo_survey_groups, evo_survey_proband) entfallen im Objektmodell, wobei diese Attribute mit Hilfe der Variablen in der *“EvoSurvey“*-Klasse (Abb. 4.4) *”configurations“*, *”groups“* und *”probanden“* realisiert sind.

Evolution-Komponente

Die Evolution-Komponente stellt die Kernkomponente der Anwendung dar und implementiert den EA-Algorithmus und das Produkt, im Sinne der Befragung.
Ein Produkt ist ein “MainIndividual”-Objekt und hat ein “Chromosome”-Objekt, das aus zwei “Genome”-Objekten besteht. Ein “MainIndividual”-Objekt kann eine Liste von “SubIndividual”-Objekte besitzen.

Als nächstes wird eine Übersicht der Modell-Komponenten dargestellt und kurz erklärt. Anschließend werden die Implementierungs-Details der Klassen in Unterkapitel 4.2.2 erläutert.

[1] Aus Platz gründen wurden die Convenience-Methoden weggelassen.

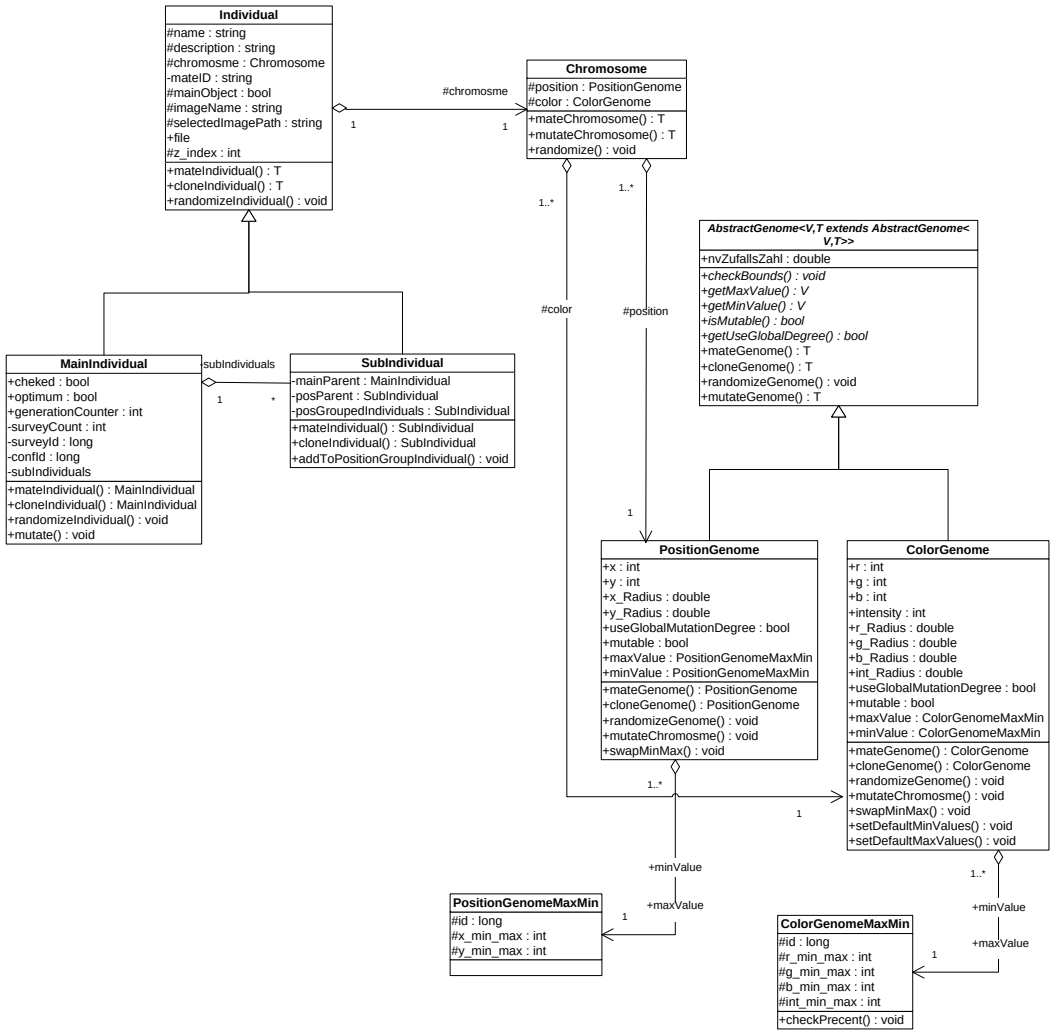


Abbildung 4.3: UML-Klassendiagramm der Evolution-Komponente

Modell-Komponente

Die Modell-Komponente stellt die für die Durchführung der Befragung notwendigen Klassen zur Verfügung. Die in UML-Klassendiagramm in Abbildung 4.4 dargestellte Klasse “EvoSession”[2] verwaltet die Zugriffe auf die Datenbank aus der Anwendungsschicht mit Hilfe der Datenbankzugriff-Schnittstelle.

[2]Die Methode *“...()”* kennzeichnet, dass die restlichen Methoden nur mit verschiedenen Objekttypen operieren.

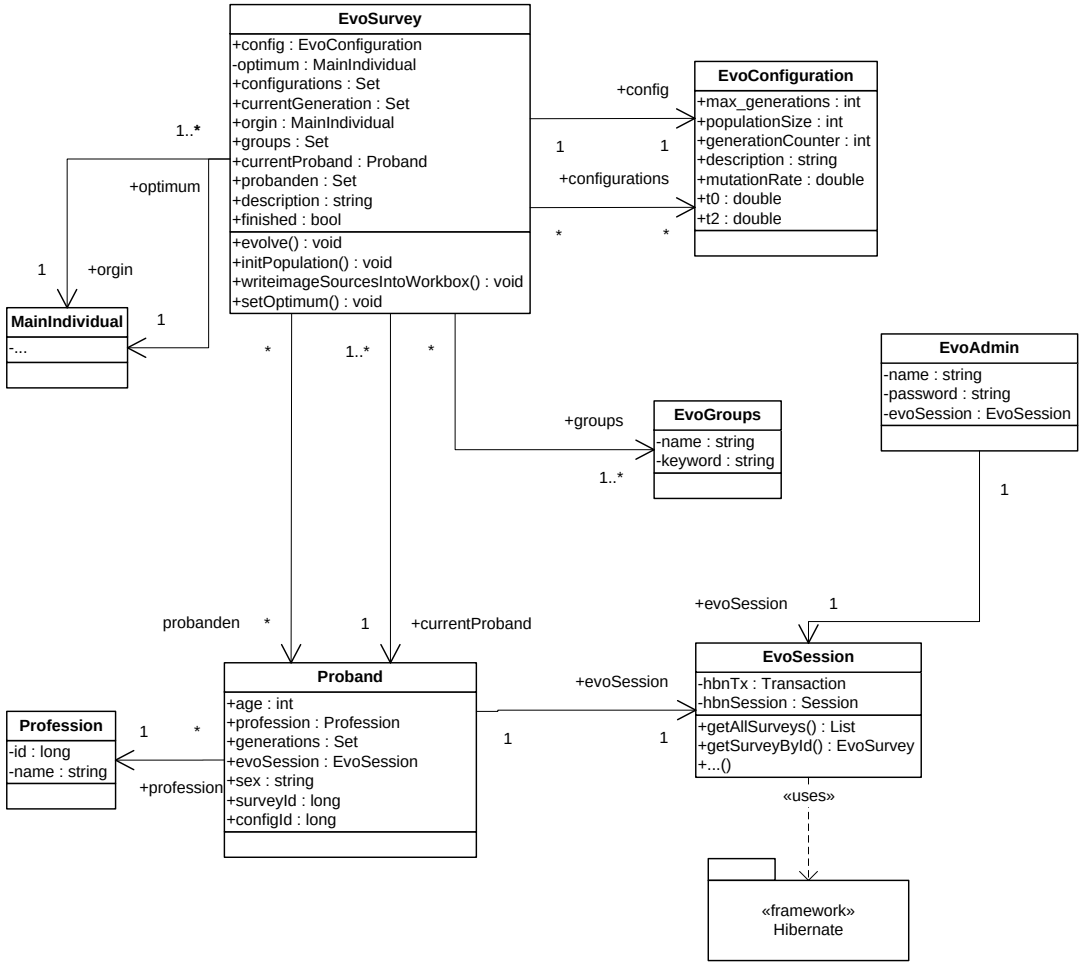


Abbildung 4.4: UML-Klassendiagramm der Modell-Komponente

4.1.2.4 Anwendungszugriff-Schnittstelle

Um die Präsentationsschicht von der Anwendungsschicht sauber zu trennen, wurde die Anwendungszugriff-Schnittstelle eingeführt. Jeder Zugriff aus der Präsentationsschicht auf die Anwendungsschicht erfolgt über diese Schnittstelle. Damit kann die Anwendungsschicht unabhängig von der Präsentationsschicht weiterentwickelt und erweitert werden. Die Anwendungszugriff-Schnittstelle wurde durch den Einsatz der Klassen, die von dem “Struts2“-Framework bereitgestellten Schnittstellen implementieren, umgesetzt. Diese Klassen enthalten zusammen mit ”JSP“-Seiten auch die Darstellungslogik.

4.1.2.5 Präsentationsschicht

Weil es sich bei der EvoMarketOnline-Anwendung um eine Web-Applikation handelt, erfolgt die Bedienung der Anwendung über Webseiten mittels eines Web-Browsers. Dabei übernehmen die

Java Web-Technologie “JSP” zusammen mit “Struts2”-Tags[3] die Aufgabe, die Webseiten darzustellen.
Bei der Entwicklung von Benutzerschnittstellen ist die Berücksichtigung der späteren Nutzergruppe von hoher Bedeutung. In diesem Fall wird anhand der Annahme, dass der typische Nutzer der EvoMarketOnline-Anwendung den Umgang mit Webseiten bereits gewohnt ist, ein Grundverständnis der Nutzer für den Umgang mit Webseiten vorausgesetzt.

4.2 Realisierung

In diesem Unterkapitel wird die Realisierung der EvoMarketOnline-Anwendung beschrieben. Nachdem im ersten Teil auf Vorgehen der Implementierung eingegangen wurde, folgt eine Erläuterung der Implementations-Schritte der oben erläuterten Komponenten.

4.2.1 Vorgehen

Die Entwicklung der EvoMarketOnline-Anwendung erfolgte in mehreren Schritten, die aufeinander aufbauen und sich gegenseitig beeinflussen. Der Entwicklungsprozess begann mit der Untersuchung der Anforderungen und mit der Analyse der Implementation der vorhandenen Lösung “Evomarket”.
Darauf aufbauend wurde die Architektur der Anwendung erstellt und anschließend die Anwendung implementiert. Nach einer Überprüfung der Anforderungen wurde die Anwendung online gestellt.
Weil einige Schritte Auswirkungen auf die vorherigen Schritte hatten, wurden die einzelnen Entwicklungsschritte nicht einzeln abgeschlossen. In dieser Hinsicht weicht der Entwicklungsprozess von dem klassischen Wasserfallmodell ab. Probleme, die z. B. erst bei oder nach der Realisierung festgestellt wurden, hatten Änderungen auf den Systementwurf zur Folge. Das Vorgehen kann insgesamt als eine leicht abgewandelte Form des evolutionären Prototypings (vgl. [FK04]) bezeichnet werden.

4.2.2 Implementation

Im Folgenden werden die Implementation-Details der Evolution-Komponenten näher erläutert. Die Details der anderen Komponenten werden weggelassen, da sie keine relevanten Details zur Evolutionären Algorithmen besitzen. Um den aktuellen Implementations-Stand der EvoMarketOnline-Anwendung zu verdeutlichen, werden die Implementations-Details aus “Evomarket”, falls vorhanden, möglichst mit der aktuellen Implementierung verglichen.

Eines der Ziele dieser Arbeit war, die bestehende Lösung so zu erweitern, dass der eingesetzte Algorithmus auf beliebige Produkte anwendbar ist. Wie bereits in der Schwachstellenanalyse (siehe 3.2) erwähnt, bietet die bestehende Lösung kaum die Möglichkeit ein anderes Produkt

[3] http://struts.apache.org/2.x/docs/struts-tags.html

außer der Rumflasche einzusetzen.
Um die entstehende Lösung möglichst universal zu halten, wurde für die Implementierung des Produktes ein objektorientiertes Modell eingesetzt.
Das bereits in Abbildung 4.3 dargestellte Modell besteht aus drei Haupt-Klassen (“Individual”, “Chromosome” und “AbstractGenome”) und zwei Hilfs-Klassen (“PositionGenomeMaxMin” und “ColorGenomeMaxMin”).

Die Klassen “MainIndividual” und “SubIndividual” wurden von der Klasse “Individual” abgeleitet, wobei die Objekte der Klasse “MainIndividual” die Basis eines Produktes darstellen, die Objekte der Klasse “SubIndividual” dagegen bilden die Teilprodukte. In den ersten Entwürfen war diese Unterteilung nicht vorhanden und wurde nach den ersten Überprüfungen eingeführt. Der Grund hierfür war, dass jedes Teilprodukt eine potenzielle Basis für das Hauptprodukt bilden konnte, was u. a. erhebliche Probleme bei dem Zusammensetzen der Bilder zur Folge hatte. Jedes “Individual”-Objekt hat ein “Chromosome”-Objekt, dies wiederum besitzt ein “ColorGenome”-Objekt und ein “PositionGenome”-Objekt.

Nachdem eine Befragung erzeugt und ein Hauptprodukt (“orgin”) zugewiesen wurde, läuft die Evolution der Produkte in der EvoMarketOnline-Anwendung folgendermaßen ab:

1. **Initialisierung:** Initialisieren der Startpopulation.
2. **Präsentation:** Darstellen der Startpopulation auf der Webseite.
3. **Evolution:**Aus den von den Probanden ausgewählten *(Selektion)* Produkten die neue Population erzeugen *(Rekombination und Mutation)*.
4. **Präsentation** Die neue Generation auf der Webseite repräsentieren.

Die letzten zwei Schritte werden solange wiederholt, bis eine Abbruchbedingung erfüllt ist.

Die Implementierung-Details werden anhand der oben genannten Schritten bzw. Teilschritten “Initialisierung”, “Rekombination” und “Mutation” erläutert. Die Selektion entspricht der Auswahl der Produkte durch die Probanden, die Präsentation enthält keine EA-relevanten Informationen, weshalb diese beiden Schritte nicht mehr behandelt werden.

Initialisierung

Bei der Initialisierung wird erst das Hauptprodukt mit Teilprodukten “n” Mal vervielfacht, wobei “n” die vorgegebene maximale Population Zahl ist, dann werden die Gen-Werte der Haupt- und Teilprodukte mit den zufälligen Zahlen besetzt.

Weil die Produkte letztendlich die Instanzen der Klasse “Individual“ sind, werden die für die Vervielfachung und für die zufällige Besetzung der Gen-Werte nötigen Methoden von der Klasse ”Individual“ und deren Unterklasse bereitgestellt.

Java bietet standardmäßig nur das flache Kopieren [4] an. Deshalb wurde das sogenannte “Deep-Copy”-Verfahren durch die Implementierung der Methode *“cloneIndividual()”* realisiert.
Listing 4.1 zeigt die Implementierung dieser Methode in den Klassen “Individual”, “MainIndividual” und “SubIndividual”.

[4] Dies bedeutet, dass Referenzen auf Objekte, die von dem zu klonenden Objekt ausgehen, beibehalten und diese Objekte nicht extra kopiert werden. [Ull07, Seite 831]

Listing 4.1: Individual Klonen

```
1  //public class  Individual ...
2  protected <T extends Individual> T cloneIndividual(Class<T> clazz) {
3      T  result = null;
4      ByteArrayOutputStream baos = new ByteArrayOutputStream();
5      try {
6          new ObjectOutputStream(baos).writeObject(this);
7          ByteArrayInputStream bais = new ByteArrayInputStream(baos.toByteArray
               ());
8          result = (clazz.cast(new ObjectInputStream(bais).readObject()));
9      } catch (IOException e) {
10         e.printStackTrace();
11     } catch (ClassNotFoundException e) {
12         e.printStackTrace();
13     }
14     result.id=TempIdGenerator.getNextFromDB();
15     return result;
16 }
17 //public class MainIndividual ...
18 public MainIndividual cloneIndividual(){
19     MainIndividual result=super.cloneIndividual(MainIndividual.class);
20     List<SubIndividual> clonedIndividuals=new ArrayList<SubIndividual>();
21     for(SubIndividual si:result.getSubIndividuals()){
22         SubIndividual clonedSi=si.cloneIndividual();
23         clonedIndividuals.add(clonedSi);
24     }
25     result.resetSubindividualsAndAddIndividuals(clonedIndividuals);
26     return result;
27 }
28
29 //public class SubIndividual ...
30 public SubIndividual cloneIndividual(){
31     SubIndividual result=super.cloneIndividual(SubIndividual.class);
32     if(result.getMainParent()!=null)
33         result.getMainParent().removeSubIndividual(result.getId()+"");
34
35     result.positionGroupIndividuals=null;
36     if(this.hasGroupedIndividuals()){
37       for(SubIndividual si:this.getPositionGroupIndividuals()){
38           result.addToPositionGroupIndividuals(si.cloneIndividual());
39       }
40     }
41     return result;
42
43 }
```

Während der Implementierung der EvoMarketOnline-Anwendung wurde die generische Java Programmierung “Java Generics” häufig eingesetzt, um den Programmierungsaufwand zu reduzieren.

Die Methode *“cloneIndividual()”* in der Klasse “Individual” erwartet als Parameter, anders als in den Unterklassen, eine Class-Instanz der aufrufenden Unterklassen. Diese Methode serialisiert das aufrufende Objekt (Zeile 6) und deserialisiert (Zeile 7 und 8) sie danach gleich wieder. Darauffolgend weist die Methode der neuen Instanz eine neue Id zu (Zeile 14), damit das Objekt in der Anwendung eindeutig identifizierbar ist. Anschließend wird die neue Instanz zurückgegeben. Die Unterklassen erweitern bzw. ergänzen diese Methode, indem sie erst die Methode der Oberklasse aufrufen (Zeile 19 und 31) und dann die Listen neu besetzen und die Abhängigkeiten wiederherstellen.
Das Gegenstück dieser Methode in “Evomarket” zeigt Listing 4.2. Weil in “Evomarket“ die Gene als einfaches Integer-Array (“int []”) und nicht als Objekte realisiert sind, werden lediglich die alten Werte in das neue Array kopiert.

Listing 4.2: Klonen in Evomarket

```
1  public class Chromosome {
2  private int[] genes;
3      Chromosome() {
4          genes = new int[numberOfGenes];
5      }
6  ...
7  Chromosome cloneChromosome(Chromosome mother) {
8      Chromosome clone = new Chromosome();
9      for (int i = 0; i < numberOfGenes; i++) {
10         clone.genes[i] = mother.getGene(i);
11     }
12     return clone;
13 }
14 ...
15 }
```

Nachdem ein Produkt geklont wurde, müssen alle Gen-Werte mit den zufälligen Zahlen besetzt werden. Dies geschieht beim Aufruf der Methode “ranomizeIndividual()”. Diese Methode aus der Klasse “MainIndividual” ruft die Methode aus der “Chromosome“-Klasse auf, die diesen Aufruf wiederum an die konkreten ”Gen“-Klassen delegiert.
Zum besseren Verständnis dieser und der nachfolgenden Methoden muss der Aufbau der konkreten ”Gen“-Klassen genauer erklärt werden.
Die konkreten ”Gen“-Klassen, in diesem Fall ”PositionGenome“ und ”ColorGenome“, sind Unterklassen der ”AbstractGenome“-Klasse. Diese abstrakte Klasse implementiert alle Evolution-Methoden. Die konkreten Unterklassen müssen bei der Deklaration zwei Parameter haben, der erste Parameter ist die ”MaxMinValue“-Klasse *(”V“)*, die die Grenzen der Gen-Werte definiert, der Zweite ist die konkrete Klasse selbst *(”T“)*. Zur Veranschaulichung sehen dann die Deklarationen der ”ColorGenome“ und ”PositionGenome” folgendermaßen aus.

```
public class ColorGenome extends AbstractGenome<ColorGenomeMinMax, ColorGenome
    > {
..............
}
```

```
public class PositionGenome extends AbstractGenome<PositionGenomeMaxMin,
    PositionGenome>{
. . . . . .
}
```

Somit stehen die Typ-Informationen über die konkreten Klassen und deren “MaxMinValue“-Klassen für die AbstractGenome“-Klasse zur Verfügung. Das allein reicht aber nicht aus, um die relevanten Methoden für die Evolution unabhängig von den konkreten Gen-Klassen zu implementieren. Dazu müssen auch die Informationen über die Felder und Methoden, welche die Gen-Werte enthalten, zur Verfügung stehen. Hierfür wurden Metadaten, sogenannte ”Annotationen“, eingesetzt. Die EvoMarketOnline-Anwendung verfügt über zwei Annotation-Klassen (Listing 4.3), ”GenomeMutableField“ und ”GenomeMutableMethod“, mit deren Hilfe die Felder und die Methoden als Gen-Werte gekennzeichnet werden können.

Listing 4.3: Annotationen in EvoMarketOnline-Anwendung

```
@Target({ElementType.FIELD})
@Retention(RetentionPolicy.RUNTIME)
public @interface GenomeMutableField {
    String mutateMethodName() default "";
    String randomMethodName() default "";
    String mutationDegreeField() default "";
    String min_maxField();}
@Target({ElementType.METHOD})
@Retention(RetentionPolicy.RUNTIME)
public @interface GenomeMutableMethod {
    String targetMethodName();
    Class<?> targetMethodParam();
    String randomMethodName() default "";
    String mutationDegreeField() default "";
    String min_maxField();}
//Benutzung (aus der Klasse ColorGenome)
@GenomeMutableField(mutationDegreeField="r_Radius", min_maxField = "r_min_max
    ")
public int r;
```

Die Verwendung der Annotationen ermöglicht es, sowohl die Felder als auch die Methoden einfach als Gen-Werte zu markieren und diese im EA-Algorithmus zu bearbeiten.
Die vollständige Unabhängigkeit des EA-Algorithmus von den konkreten Gen-Klassen wird aber erst durch auslagern der Methoden in eine abstrakte Klasse und durch den Einsatz von ”Java Reflections“ erreicht.
Nachdem der Aufbau der konkreten ”Gen“-Klassen erläutert wurde, wird als nächstes die Methode ”randomizeGenome()“ erklärt. Weil die Methode sehr lang ist, wird im Listing 4.4 nur ein Ausschnitt dargestellt. Der vollständige Quelltext der Klasse ist im Anhang (A.1) zu finden.

Listing 4.4: Die Gen-Werte mit zufälligen Werten belegen

```
1 public void randomizeGenome(Class<T> clazz, EvoConfiguration config){
2     double genValue=0;
3     double currentMaxVariance=0;
```

```
4       double currentMinVariance=0;
5       for (Field f: clazz.getDeclaredFields()){}
6       GenomeMutableField annotation = f.getAnnotation(GenomeMutableField.class)
            ;
7       if (annotation != null) {
8        String methodName = annotation.randomMethodName();
9       //wenn keine spezielle Random-Methode angegeben wurde
10      if (methodName.equals("")) {
11      if (f.get(this) instanceof Number) {
12          //weil double immer richtig ist
13          f.setAccessible(true);
14          genValue=f.getDouble(this);
15          //Ueberpruefen obere und untere Grenzen
16          if(getMaxValue()!=null){
17          currentMaxVariance=(getMaxValue().getClass().getDeclaredField(
                annotation.min_maxField())).getDouble(getMaxValue());
18          }
19          if(getMinValue()!=null){
20          currentMinVariance=(getMinValue().getClass().getDeclaredField(
                annotation.min_maxField())).getDouble(getMinValue());
21          }
22          if (config.getMutationRate() > Math.random()) {
23              double range;
24              range = currentMaxVariance-currentMinVariance;
25              genValue=currentMinVariance +  (int) (range * Math.random());
26          }
27          f.set(this, Number.class.getDeclaredMethod( f.getType().getName()+"
                Value").invoke(genValue));
28        }else {
29         logger.error("Diese Methode kann nur mit Nummern umgehen");
30         }
31      }else{
32       //wenn spezielle Mutation-Methode angegeben wurde, fuehre die spezielle
             Methode aus
33        Method mutator= clazz.getDeclaredMethod(methodName, null);
34        mutator.setAccessible(true);
35        mutator.invoke(this);
36       }
37      }
38      //Dieselbe Prozedur fuer Methoden der Klasse
39  ...
40
41  }
```

Wie man dem Listing entnehmen kann, operiert die Methode ”randomizeGenome()“ völlig unabhängig von den konkreten Unterklasse. Zum Vergleich ist in Listing 4.5 die entsprechende Methode aus ”Evomarket“, die auch in 4.4 (Zeilen 23 bis 25) vorkommt, zu sehen.

Listing 4.5: Die Gen-Werte mit zufälligen Werten belegen in Evomarket

```
1 public void randomizeGene(int i) {
2     double range;
3     range = maxVariance.getGene(i) - minVariance.getGene(i);
4     setGene(i, minVariance.getGene(i) + (int) (range * Math.random()));
5 }
```

Der Wunsch nach der Unabhängigkeit der Methoden von den konkreten Klassen führt zu einer aufwändigeren Implementierung. Dieser Aufwand wurde dennoch in Kauf genommen, weil diese Implementierung, neben der Unabhängigkeit von den konkreten Klassen

- die Flexibilität der Gen-Klassen erlaubt,
- die Erweiterbarkeit der Anwendung vereinfacht, und
- die Möglichkeit bietet, den angewandten Algorithmus ohne großen Aufwand auszutauschen.

Alle Methoden der Klasse ”AbstractGenome” wurden nach diesem Schema implementiert, sie unterscheiden sich nur in der Funktionalität.
Nachdem die Gen-Werte initialisiert wurden, werden die Bilder der Haupt- und Teilprodukte zusammengesetzt und auf dem Web-Browser präsentiert. Die Probanden haben die Möglichkeit, ein oder zwei Produkte als Eltern für die nächste Generation auszuwählen. In dem Fall, dass zwei Produkte als Eltern ausgewählt werden, müssen die Gen-Werte der beiden Produkte rekombiniert werden.

Rekombination

Wie in den anderen wichtigen Operationen, die für die Evolution wichtig sind, wurde die Rekombination auch in der Klasse ”AbstractGenome” realisiert. Diese Implementierung ähnelt stark der Methode ”randomGenome”, die bereits zuvor erklärt wurde. Ein Unterschied ist die Berechnung des Durchschnittswertes der Gen-Werte (Zeilen 14 und 22).
Die hier eingesetzte Rekombination entspricht der intermediären Rekombination aus Abschnitt 2.2.2.1.

Listing 4.6: Rekombination

```
1
2 protected <T extends Individual >T mateIndividual(T anotherIndiVidual, Class<
      T> clazz) {
3 ...
4 if (fieldAnno != null) {
5 if (f.get(this) instanceof Number) {
6     f.setAccessible(true);
7     selfGenValue = f.getDouble(this);
8     if (anotherGenome != null) {
9         genValueToMate = (anotherGenome.getClass()
10                .getDeclaredField(f.getName()))
11                .getDouble(anotherGenome);
12    }
```

```
13      Field targetField=result.getClass().getDeclaredField(f.getName());
14      double targetValueToSet=((selfGenValue + genValueToMate) / 2);
15      Method mathMeth=Number.class.getDeclaredMethod( targetField.getType().
            getName()+"Value");
16      targetField.set(result,  mathMeth.invoke(targetValueToSet));
17      String mutatDegree=fieldAnno.mutationDegreeField();
18      if(mutatDegree!=null||!"".equals(mutatDegree)){
19          Field selfDegreeField=clazz.getField(mutatDegree);
20          double degreeValueToMate = (anotherGenome.getClass().getDeclaredField
                (selfDegreeField.getName())).getDouble(anotherGenome);
21          double selfDegreeValue= selfDegreeField.getDouble(this);
22          double matedValue=((degreeValueToMate + selfDegreeValue) / 2);
23          Field matedField=result.getClass().getDeclaredField(selfDegreeField.
                getName());
24          mathMeth=Number.class.getDeclaredMethod( matedField.getType().getName
                ()+"Value");
25          matedField.set(result,  mathMeth.invoke(matedValue));
26      }
27
28  } else {
29  logger.error("Diese Methode kann nur mit Nummern umgehen");
30  }
31  }
32
33
34  ...
35  }
```

Mutation

In der Methode “mutate” wird der Mutation-Operator implementiert. Sie stellt die wichtigste Komponente der Evolution dar. Der Operator entspricht dem EA-Mutation-Operator in Grundform aus Abschnitt 2.2.2.1. Zur besseren Veranschaulichung wird die Formel des Operators an dieser Stelle erneut aufgeführt.

Standardabweichung:

Falls nur mit einer Standardabweichung gearbeitet wird :

$$\acute{\sigma} = \sigma.exp(\tau_0.N(0,1)) \tag{1}$$

Für jedes Kind (K) *eine separate Standardabweichung:*

$$\acute{\sigma_k} = \sigma_k.exp(\tau_1.N(0,1) + \tau_2.N_k(0,1)) \tag{2}$$

Die Strategieparameter τ_1 *und* τ_2 *sind dabei exogene Konstante.*

Mutation der Kindesvariablen: (3)

$$\acute{x_j} = x_j + \acute{\sigma}.N_j(0,1) \qquad j = (1,2,..,n) \text{ und } (\forall \geq n_\sigma : \acute{\sigma_j} = \acute{\sigma_{n_\sigma}})$$

Die Zeile 36 in Listing 4.7 entspricht der Formel 1, wobei tempDegree die Standardabweichung μ und nvZufallsZahl $N(0,1)$ darstellt. Die Formel 2 findet sich in Zeile 34 wieder, Formel 3 in Zeile 40.

Listing 4.7: Mutation

```
1  public AbstractGenome() {
2          super();
3          Random rnd=new Random();
4          this.nvZufallsZahl=rnd.nextGaussian();
5
6      }
7  public void mutateGenome(EvoConfiguration config){
8  Random random=new Random();
9  if(this.isMutable()){
10     Class<?> cl = this.getClass();
11     try {
12     double genValue=0;
13 for (Field f: cl.getDeclaredFields())
14 {
15     GenomeMutableField annotation = f.getAnnotation(GenomeMutableField.class)
           ;
16     if (annotation != null) {
17     String methodName = annotation.mutateMethodName();
18     String mutationDegreeField=annotation.mutationDegreeField();
19     if (methodName.equals("")) {
20     if (f.get(this) instanceof Number) {
21         f.setAccessible(true);
22         genValue=f.getDouble(this);
23         if (config.getMutationRate() > Math.random()) {
24             double tempDegree=0.0;
25                 Field degreeField=null;
26             if(mutationDegreeField==null||"".equals(mutationDegreeField)||
                   getUseGlobalMutationDegree()){
27                 tempDegree=config.getMutationDegree();
28             }else{
29                 degreeField=cl.getField(mutationDegreeField);
30                 degreeField.setAccessible(true);
31                     tempDegree = degreeField.getDouble(this);
32             }
33             if(config.t2!=0.0){
34             tempDegree=tempDegree*Math.exp(config.t0*random.nextGaussian()+
                   config.t2*nvZufallsZahl);
35             }else{
36                 tempDegree=tempDegree*Math.exp(config.t0*nvZufallsZahl);
37             }
38
```

```
39
40              genValue=genValue + tempDegree*random.nextGaussian();
41
42
43          }
44          f.set(this, Number.class.getDeclaredMethod( f.getType().getName()+"
                Value").invoke(genValue));
45          }else {
46           logger.error("Diese Methode kann nur mit Nummern umgehen");
47          }
48      }else
49      {
50      Method mutator= cl.getDeclaredMethod(methodName, null);
51      mutator.setAccessible(true);
52      mutator.invoke(this);
53      }
54      }
55  }
56
57  for(Method m:cl.getDeclaredMethods()){
58  ...
59  }
60  ...
61  }
62  ...
63  }
```

Listing 4.8 stellt die Mutation-Operation in Evomarket dar. Der Vergleich der beiden Listings verdeutlicht, dass die Mutation-Operationen sich voneinander stark unterscheiden.

Listing 4.8: Mutation in Evomarket

```
1  public void mutateGene(int i) {
2      if (mutationRate > Math.random()) {
3          mutationRange = (maxVariance.getGene(i) - minVariance.getGene(i))*
               mutationDegree;
4          int value=getGene(i) + (int) (mutationRange / 2 * (Math.random() -
               0.5));
5          setGene(i, value);
6      }
7  }
```

Der in der EvoMarketOnline-Anwendung implementierter Mutation-Operator, wie bereits erwähnt, entspricht dem EA-Mutation-Operator in Grundform. Der in Evomarket eingesetzter Operator verwendet, entgegen der Literatur, [5] eine gleichverteilte Zufallszahl [Jav] zur Bestimmung der neuen Gen-Werte. (Listing 4.8 Zeile 4)
Dies hat in Kombination mit der “mutationDegree“-Variablen zur Folge, dass die neuen Werte sich in bestimmten Intervallen bewegen. Wie z.B. eine Mutationsschrittweite (*“mutationDe-*

[5] Vgl. [SHF94, Seite 174], [GKK04, Seite 119], [Nis97, Seite 157]

gree“) von 0.8 entspricht einer maximale Änderung von $\pm 20\%$ der möglichen Mutationsspanne (”maxVariance-minVariance”):
Der Ausdruck *“(Math.random() - 0.5)”* kann nur die Werte zwischen -0.5 und $+0.5$ liefern. In der folgenden Tabelle werden die möglichen maximalen Änderungen für die Mutationsspannen “30”,“50”, “60‘ und ”100“ aufgeführt.

Mutationsspanne	**Berechnung**	**Ergebnis**
30	$((30 * 0.8)/2) * \pm 0.5 = \pm 6$	$\pm 6 = \pm 20\%$ von 30
50	$((50 * 0.8)/2) * \pm 0.5 = \pm 10$	$\pm 10 = \pm 20\%$ von 50
60	$((60 * 0.8)/2) * \pm 0.5 = \pm 12$	$\pm 12 = \pm 20\%$ von 60
100	$((100 * 0.8)/2) * \pm 0.5 = \pm 20$	$\pm 20 = \pm 20\%$ von 100

Zur Bestimmung der neuen Gen-Werte sollte eine normalverteilte Zufallszahl mit dem Erwartungswert “0” eingesetzt werden, damit sich die Gen-Werte, wie in der Natur, beliebig ändern können. Um die Auswirkungen der beiden Verteilungen auf die Gen-Werte zu verdeutlichen, wurde ein kleiner Test mit einer Mutationsspanne von “10” und einer Schrittweite “0.8” ausgeführt. Der Quelltext dieses Tests und die zugehörigen Tabellen sind im Anhang A.2 zu finden. Darüber hinaus kann das Heranziehen der Mutationsspanne die Konvergenz der Produkte, deren Gen-Werte extrem unterschiedliche Mutationsspannen haben, erschweren.
Zur Veranschaulichung wird im Folgenden angenommen, dass zwei Gene (“R” und “G”) eine Mutationsspanne von “10” und “100” haben. Die Mutationsschrittweite beträgt wie vorher “0.8”. Wie oben erwähnt eine Mutationsschrittweite von “0.8” entspricht einer maximale Änderung von $\pm 20\%$ der möglichen Mutationsspanne. Daraus ergibt sich für “R” eine maximale Änderung von ± 2-Einheiten und für “G” eine maximale Änderung von ± 20-Einheiten. Dies kann zur Folge haben, dass das Gen “G” dominant bleibt und damit die Konvergenz erschwert.

5 Resümee

Dieses Kapitel beginnt mit einem Vergleich zwischen der vorhandenen Lösung und der entwickelten “EvoMarketOnline“. Anschließend werden die erreichten Ziele dieser Arbeit in einem Fazit diskutiert. Zum Abschluss wird es dann einen kurzen Ausblick geben. Er soll mögliche Schritte für eine Weiterentwicklung der ”EvoMarketOnline”-Anwendung aufzeigen.

5.1 Fazit

Ziel dieser Arbeit war es u. a., einen Entwurf für eine Software für Online-Befragungen mit interaktiven evolutionären Algorithmen zu machen und anschließend diesen Entwurf zu implementieren.
Die Frage, die an dieser Stelle beantwortet werden soll, lautet: Sind bei dem Entwurf und bei der Implementierung alle Anforderungen an die EvoMarketOnline berücksichtigt worden? In dem Unterkapitel 3.3 wurden die Anforderungen an die Anwendung aufgezählt. Dabei wurde eine Unterteilung in fachliche und technische Anforderungen vorgenommen.

5.1.1 Erfüllung der fachlichen Anforderungen

Die fachlichen Anforderungen wurden von dem Leiter des Arbeitsbereiches Marketing und Innovation, Prof. Dr. Thorsten Teichert, geprüft. Deshalb wird auf dieses Thema nicht mehr eingegangen. Die Ergebnisse dieser Überprüfung ist im Anhang A.3 zu finden.

5.1.2 Erfüllung der technischen Anforderungen

Die Realisierung der technischen Anforderungen ist die Vorbedingungen für die Überprüfung der fachlichen Anforderungen.

Wie bereits erwähnt wurden die fachlichen Anforderungen von dem Arbeitsbereich Marketing und Innovation der Universität Hamburg überprüft, weshalb bezüglich der technischen Anforderungen gesagt werden kann, dass diese erfüllt wurden. Darüber hinaus durch den Einsatz von Tomcat und “Java EE 5”[1] sind keine Mehrkosten für den Arbeitsbereich Marketing und Innovation der Universität Hamburg entstanden.

[1] http://java.sun.com/javaee/technologies/javaee5.jsp

Die für die Persistenz eingesetzte Technologien, die relationale Datenbank ”MySQL“[2] und das Persistenz-Framework ”Hibernate“[3], sind Open-Source und verursachen ebenfalls keine Mehrkosten.
Vorschläge für eine mögliche Weiterentwicklung und Erweiterung der EvoMarketOnline folgen im nächsten Abschnitt.

5.2 Ausblick

Abschließend werden hier in einem Ausblick einige Möglichkeiten für eine Weiterentwicklung der EvoMarketOnline vorgeschlagen.

Bei dem Entwurf und der Realisierung der EvoMarketOnline-Anwendung wurde auf die einfache Erweiterbarkeit großen Wert gelegt, was zur Folge hat, dass der in Unterkapitel 3.4 beschriebene Soll-Zustand ohne großen Programmierungsaufwand erreicht werden kann. Das Erreichen des Soll-Zustandes ist allerdings mit einem Zeitaufwand verbunden. Die unter 4.2.2 eingesetzte Möglichkeiten von Java ”Generics, Annotationen“ und ”Reflections“ ermöglichen es, dies zu erreichen.
Darüberhinaus kann EvoMarketOnline so erweitert werden, dass sie ein Produkt dreidimensional darstellen kann. Dafür muss die Präsentationsschicht der Anwendung mit einer geeigneter Technologie wie z. B. ”JNLP“[4] ausgetauscht werden, was dank der losen Kopplung der Anwendungsschichten kein großes Problem darstellen sollte.
Zum Schluss sei folgendes angemerkt. Die EvoMarketOnline-Anwendung ist zum gegenwärtigen Zeitpunkt voll funktions- und einsatzfähig. Sollte der Wunsch nach zusätzlichen Funktionen bestehen, kann über eine Weiterentwicklung nachgedacht werden. Die oben genannten Punkte und der Soll-Zustand sollten dafür als Anregung verstanden werden.

[2] http://www.mysql.de/
[3] http://www.hibernate.org/
[4] http://java.sun.com/products/javawebstart/

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Literaturverzeichnis

[aff] Procter & Gamble - Tampax®Case Study. http://www.flagshipventures.com/presentations/affinnova/affinnova2.pdf - Abruf am 11.11.2007.

[BEPW05] Klaus Backhaus, Bernd Erichson, Wulff Plinke, and Rolf Weiber. *Multivariate Analyseverfahren*. Springer, 2005.

[Dar59] Charles Darwin. *The Origin of Species*. John Murray, 1859.

[DH07] Jürgen Dunkel and Andreas Holitschke. *Softwarearchitektur für die Praxis*. Springer Verlag, 2007.

[FK04] Peter Forbrig and Immo Kerner. *Lehr- und Übungsbuch Softwareentwicklung*. Hanser Fachbuchverlag, 2004.

[GHJV01] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Entwurfsmuster - Elemente wiederverwendbarer objektorientierter Software*. Addison-Wesley, 2001.

[GKK04] Ingrid Gerdes, Frank Klawon, and Rudolf Kruse. *Evolutionäre Algorithmen*. vieweg, 2004.

[Jav] Random (Java 2 Platform SE v1.4.2). http://java.sun.com/j2se/1.4.2/docs/api/java/util/Random.html - Abruf am 10.02.2008.

[Mef00] Heribert Meffert. *Marketing*. Gabler, 2000.

[NHB+98] Volker Nissen, Martin Tietze (Hrsg.), Jörg Biethahn, Albrecht Hönerloh, Jochen Kuhl, and Marie-Claire Leisewitz. *Betriebswirtschaftliche Anwendungen des Soft Computing*, chapter Einige Grundlagen Evolutionärer Algorithmen, pages 55–78. vieweg, 1998.

[Nis94] Volker Nissen. *Evolutionäre Algorithmen*. DeutscherUniversitätsVerlag, 1994.

[Nis97] Volker Nissen. *Einführung Evolutionäre Algorithmen*. Vieweg, 1997.

[Rec] Prof. Dr.-Ing. Ingo Rechenberg. Evolutionistische Bionik auf dem Prüfstand. Der Fundamentalbeleg der Bionik. http://www.bionik.tu-berlin.de/institut/skript/B1Fol2.ppt - Abruf am 14.11.2007.

[Sch05] Karl Scharbert. *Requirements Analysis realisieren*. vieweg, 2005.

[SGHE07] Thorsten Teichert & Edlira Shehu, Anders Gustafsson, Andreas Herrmann, and Huber Frank (Eds.). *Conjoint Measurement Methods and Applications*, chapter Evolutionary Conjoint, pages 113–130. Springer, 2007.

[SHF94] Eberhard Schöneburg, Frank Heinzmann, and Sven Feddersen. *Genetische Algorithmen und Evolutionsstrategien*. Addison-Wesley, 1994.

[Sta05] Thomas Stark. *J2EE Master Class. Einstieg für Anspruchsvolle*. Pearson Studium, 2005.

[THH00] Thorsten Teichert, Andreas Herrmann, and Christian Homburg (Hrsg.). *Handbuch Marktforschung, Methoden - Anwendungen - Praxisbeispiele*, chapter Conjoint-Analyse, pages 471–512. Gabler (Wiesbaden), 2000.

[uE] Technische Universitäat Berlin FG Bionik und Evolutionstechnik. Prof. Dr.-Ing. Ingo Rechenberg. http://www.bionik.tu-berlin.de/institut/n2rechenb.html - Abruf am 10.11.2007.

[Ull07] Christian Ullenboom. *Java ist auch eine Insel*. Galileo Computing, 2007.

[Wei02] Karsten Weicker. *Evolutionäre Algorithmen*. Tuebner, 2002.

[WWS05] Martin Welker, Andreas Werner, and Joachim Scholz. *Online-Research*. dpunkt.verlag, 2005.

Symbolverzeichnis

λ	Anzahl der Nachkommen
μ	Populationsgröße
Φ	Wahrscheinlichkeitsdichte
ρ	Anzahl der Individuen, die rekombinert werden sollen
σ	Standardabweichung

Glossar

Annotationen

Ein Mittel zur Strukturierung von Quelltext, das die Einbindung der Metadaten in den Quelltext erlaubt.

Applikation-Server

Ein Server in einem Computernetzwerk, auf dem eine spezielle Software-Applikation läuft.

BLOB

Abkürzung für *Binary Large Objects* . Der Datentyp für binär gespeicherte Daten in einer "Datenbank Management System".
Wird häufig u. a. für die Speicherung der multimedialen Inhalten verwendet.
Die Unterstützung für diesen Datentyp ist nicht standardisiert.

Einplatzsystem

Datenverarbeitungssystem, das eigenständig und zu einer Zeit nur für einen Benutzer arbeitet, meist im multifunktionalen Einsatz, auch an Kommunikationsdienste anschließbar.

Flop

Bezeichnet einen wirtschaftlichen Misserfolg.

JAVA

Java ist eine objektorientierte Programmiersprache und als solche ein eingetragenes Warenzeichen der Firma Sun Microsystems. Sie ist eine Komponente der Java-Technologie.

Java EE 5

Abkürzung für *Java Plattform Enterprise Edition 5*. Ein von der Firma Sun Microsystems definierter Standard für die Entwicklung von mehrschichtigen Geschäftsanwendungen.

JNLP

Abkürzung für *Java Network Launching Protocol*. Eine Java-Technologie, die das Ausführen der Java-Anwendungen über Internet ermöglicht. Die Anwendungen benötigen keine zusätzlichen Software außer installierter *"Java Web Start"*, die standardmäßig bei der Java Installation mit installiert wird.

JSP

Abkürzung für *Java Server Pages*. Eine Java-Technologie, die das Erstellen von dynamischen Web-Inhalten ermöglicht.

PDF

Das Portable Document Format ist ein plattformübergreifendes Dateiformat für Dokumente.
Das Format wurde von der Firma Adobe Systems entwickelt und 1993 mit Acrobat 1 veröffentlicht.

RGB

Bezeichnet ein additives Farbmodel, dessen Grundfarben Rot, Grün und Blau sind.
In diesem Model wird eine Farbe durch den Anteil der Grundfarben in Prozent oder Bytewerten von 0 bis 255 definiert.

Abkürzungsverzeichnis

AMI	Arbeitsbereich Marketing und Innovation der Universität Hamburg
EA	Evolutionäre Algorithmen
EP	Evolutionäre Programmierung
ES	Evolutionsstrategien
GA	Genetische Algorithmen
GP	Genetische Programmierung
IEA	Interaktive Evolutionären Algorithmen
Java EE 5	Java Platform Enterprise Edition 5
JSP	Java Server Pages

A Anhang

A.1 Quellcode der Klasse AbstarctGenome

Listing A.1: AbstarctGenome

```
1  /**
2   *
3   */
4  package edu.haw.bul.evo.eva;
5
6  import java.io.ByteArrayInputStream;
7  import java.io.ByteArrayOutputStream;
8  import java.io.IOException;
9  import java.io.ObjectInputStream;
10 import java.io.ObjectOutputStream;
11 import java.io.Serializable;
12 import java.lang.reflect.Field;
13 import java.lang.reflect.InvocationTargetException;
14 import java.lang.reflect.Method;
15 import java.util.Random;
16
17 import org.apache.log4j.Logger;
18
19 import com.opensymphony.xwork2.ActionContext;
20
21 import edu.haw.bul.evo.annotations.GenomeMutableField;
22 import edu.haw.bul.evo.annotations.GenomeMutableMethod;
23 import edu.haw.bul.evo.app.EvoConfiguration;
24
25 /**
26  * @author Buelent Ulas
27  *
28  */
29 public abstract class AbstractGenome<V,T extends AbstractGenome<V,T>>
        implements Serializable{
30 private static final Logger logger = Logger.getLogger(AbstractGenome.class);
31 public abstract   V getMaxValue();
32 public abstract  V getMinValue();
33 public abstract boolean isMutable();
34 public abstract boolean getUseGlobalMutationDegree();
35 public abstract void checkBounds();
36 public double nvZufallsZahl;
```

```
37
38
39
40
41  /**
42      *
43      */
44  public AbstractGenome() {
45      super();
46      Random rnd=new Random();
47      this.nvZufallsZahl=rnd.nextGaussian();
48
49  }
50  public T mateGenomes(T anotherGenome, Class<T> clazz){
51  T result= null;
52  double selfGenValue = 0;
53  double genValueToMate = 0;
54  try {
55      result= clazz.newInstance();
56      //Felder durchsuchen
57      for (Field f : clazz.getDeclaredFields()) {
58      GenomeMutableField fieldAnno = f.getAnnotation(GenomeMutableField.class);
59      if (fieldAnno != null) {
60      if (f.get(this) instanceof Number) {
61          // weil double immer richtig ist
62          f.setAccessible(true);
63          selfGenValue = f.getDouble(this);
64          if (anotherGenome != null) {
65              genValueToMate = (anotherGenome.getClass()
66                      .getDeclaredField(f.getName()))
67                      .getDouble(anotherGenome);
68          }
69          Field targetField=result.getClass().getDeclaredField(f.getName());
70          double targetValueToSet=((selfGenValue + genValueToMate) / 2);
71          Method mathMeth=Number.class.getDeclaredMethod( targetField.getType()
                .getName()+"Value");
72
73          targetField.set(result,  mathMeth.invoke(targetValueToSet));
74          String mutatDegree=fieldAnno.mutationDegreeField();
75          if(mutatDegree!=null||!"".equals(mutatDegree)){
76              Field selfDegreeField=clazz.getField(mutatDegree);
77              double degreeValueToMate = (anotherGenome.getClass().
                    getDeclaredField(selfDegreeField.getName())).getDouble(
                    anotherGenome);
78              double seelfDegreeValue= selfDegreeField.getDouble(this);
79              double matedValue=((degreeValueToMate + seelfDegreeValue) / 2);
80              Field matedField=result.getClass().getDeclaredField(
                    selfDegreeField.getName());
81              mathMeth=Number.class.getDeclaredMethod( matedField.getType().
                    getName()+"Value");
```

```
82                  matedField.set(result,  mathMeth.invoke(matedValue));
83              }
84          } else {
85              logger.error("Diese Methode kann nur mit Nummern umgehen");
86              }
87          }
88          }
89      //Methoden durcsuchen
90      for(Method m:clazz.getDeclaredMethods()){
91      GenomeMutableMethod methAnno=m.getAnnotation(GenomeMutableMethod.class);
92      if(methAnno!=null){
93          if (m.getReturnType().getClass().isInstance(Number.class)) {
94              Method targetMethod=clazz.getDeclaredMethod(methAnno.targetMethodName
                    (), methAnno.targetMethodParam());
95              selfGenValue =    Double.parseDouble(m.invoke(this).toString());
96              genValueToMate=   Double.parseDouble(anotherGenome.getClass().
                    getDeclaredMethod(m.getName()).invoke(anotherGenome)+"");
97              double setValue= ((selfGenValue+genValueToMate)/2);
98          if(methAnno.targetMethodParam().isPrimitive()){
99              targetMethod.invoke(result, Number.class.getDeclaredMethod( methAnno.
                    targetMethodParam().getName()+"Value").invoke(setValue));
100         }
101         }
102
103     }
104
105     }
106     } catch (SecurityException e) {
107         logger.error(e);
108     } catch (NoSuchMethodException e) {
109             logger.error(e);
110     } catch (IllegalArgumentException e) {
111             logger.error(e);
112     } catch (IllegalAccessException e) {
113         logger.error(e);
114         System.out.println(e);
115     } catch (InvocationTargetException e) {
116         logger.error(e);
117         System.out.println(e);
118     } catch (NoSuchFieldException e) {
119         logger.error(e);
120         System.out.println(e);
121     } catch (InstantiationException e) {
122         logger.error(e);
123         System.out.println(e);
124     }
125     return result;
126     }
127
128     public void mutateGenome(EvoConfiguration config){
```

```
129     Random random=new Random();
130     this.nvZufallsZahl=random.nextGaussian();
131     boolean mutdegreeChanged=false;
132     if(this.isMutable()){
133     Class<?> cl = this.getClass();
134     try {
135     double genValue=0;
136     for (Field f: cl.getDeclaredFields()){
137         GenomeMutableField annotation = f.getAnnotation(GenomeMutableField.
                class);
138         if (annotation != null) {
139         String methodName = annotation.mutateMethodName();
140         String mutationDegreeField=annotation.mutationDegreeField();
141         if (methodName.equals("")) {
142         if (f.get(this) instanceof Number) {
143             f.setAccessible(true);
144             //weil double immer richtig ist
145             genValue=f.getDouble(this);
146             if (config.getMutationRate() > Math.random()) {
147             double tempDegree=0.0;
148                 Field degreeField=null;
149             if(mutationDegreeField==null||"".equals(mutationDegreeField)||
                    getUseGlobalMutationDegree()){
150                 tempDegree=config.getMutationDegree();
151             }else{
152                 degreeField=cl.getField(mutationDegreeField);
153                 degreeField.setAccessible(true);
154                     tempDegree = degreeField.getDouble(this);
155             }
156             if(config.t2!=0.0){
157             tempDegree=tempDegree*Math.exp(config.t0*random.nextGaussian()+
                    config.t2*nvZufallsZahl);
158             }else{
159                 tempDegree=tempDegree*Math.exp(config.t0*nvZufallsZahl);
160             }
161             mutdegreeChanged=true;
162
163             genValue=genValue + tempDegree*random.nextGaussian();
164             if(degreeField!=null){
165                 degreeField.set(this, Number.class.getDeclaredMethod(
                        degreeField.getType().getName()+"Value").invoke(
                        tempDegree));
166             }
167             }
168             f.set(this, Number.class.getDeclaredMethod( f.getType().getName()
                    +"Value").invoke(genValue));
169          }else {
170            logger.error("Diese Methode kann nur mit Nummern umgehen");
171            }
172         }else {
```

```
173             Method mutator= cl.getDeclaredMethod(methodName, null);
174             mutator.setAccessible(true);
175             mutator.invoke(this);
176            }
177           }
178       }
179
180       for(Method m:cl.getDeclaredMethods()){
181           GenomeMutableMethod methAnno=m.getAnnotation(GenomeMutableMethod.
                  class);
182           if(methAnno!=null){
183           if (m.getReturnType().getClass().isInstance(Number.class)) {
184               Method targetMethod=cl.getDeclaredMethod(methAnno.
                      targetMethodName(), methAnno.targetMethodParam());
185               genValue =    Double.parseDouble(m.invoke(this).toString());
186               if(getMaxValue()!=null)
187               if (config.getMutationRate() > Math.random()) {
188                       String mutationDegreeField=methAnno.mutationDegreeField()
                            ;
189                       double tempDegree=0.0;
190                       Field degreeField=null;
191
192                       if(mutationDegreeField==null||"".equals(
                            mutationDegreeField)||getUseGlobalMutationDegree()){
193                        tempDegree=config.getMutationDegree();
194                       }else{
195                       degreeField=cl.getField(mutationDegreeField);
196                       tempDegree = degreeField.getDouble(this);
197                       }
198
199                       if(config.t2!=0.0){
200                        tempDegree=tempDegree*Math.exp(config.t0*random.
                             nextGaussian()+config.t2*nvZufallsZahl);
201                       }else{
202                        tempDegree=tempDegree*Math.exp(config.t0*nvZufallsZahl);
203                       }
204
205                       mutdegreeChanged=true;
206                       genValue=genValue + tempDegree*random.nextGaussian();
207                       if(degreeField!=null)
208                       degreeField.set(this, Number.class.getDeclaredMethod(
                            degreeField.getType().getName()+"Value").invoke(
                            tempDegree));
209
210               }
211               if(methAnno.targetMethodParam().isPrimitive()){
212                   targetMethod.invoke(this, Number.class.getDeclaredMethod(
                        methAnno.targetMethodParam().getName()+"Value").invoke(
                        genValue));
213               }
```

```
214         }
215         }
216     }
217
218     } catch (SecurityException e) {
219         logger.error(e);
220     } catch (NoSuchMethodException e) {
221         logger.error(e);
222     } catch (IllegalArgumentException e) {
223         logger.error(e);
224     } catch (IllegalAccessException e) {
225         logger.error(e);
226     } catch (InvocationTargetException e) {
227         logger.error(e);
228     } catch (NoSuchFieldException e) {
229         logger.error(e);
230
231     }
232     }
233 checkBounds();
234 }
235
236 public void randomizeGenome(Class<T> clazz, EvoConfiguration config){
237 try {
238 double genValue=0;
239 double currentMaxVariance=0;
240 double currentMinVariance=0;
241 for (Field f: clazz.getDeclaredFields())
242 {
243     GenomeMutableField annotation = f.getAnnotation(GenomeMutableField.class)
        ;
244     if (annotation != null) {
245     String methodName = annotation.randomMethodName();
246     if (methodName.equals("")) {
247     if (f.get(this) instanceof Number) {
248         //weil double immer richtig ist
249         f.setAccessible(true);
250         genValue=f.getDouble(this);
251         //Ueberpruefen obere und untere Grenzen
252         if(getMaxValue()!=null){
253         currentMaxVariance=(getMaxValue().getClass().getDeclaredField(
            annotation.min_maxField())).getDouble(getMaxValue());
254         }
255         if(getMinValue()!=null){
256         currentMinVariance=(getMinValue().getClass().getDeclaredField(
            annotation.min_maxField())).getDouble(getMinValue());
257         }
258         if (config.getMutationRate() > Math.random()) {
259             double range;
260             range = currentMaxVariance-currentMinVariance;
```

```
261             genValue=currentMinVariance +  (int) (range * Math.random());
262         }
263         f.set(this, Number.class.getDeclaredMethod( f.getType().getName()+"
                Value").invoke(genValue));
264       }else {
265        logger.error("Diese Methode kann nur mit Nummern umgehen");
266        }
267     }else{
268       Method mutator= clazz.getDeclaredMethod(methodName, null);
269       mutator.setAccessible(true);
270       mutator.invoke(this);
271      }
272     }
273 }
274
275 for(Method m:clazz.getDeclaredMethods()){
276     GenomeMutableMethod methAnno=m.getAnnotation(GenomeMutableMethod.class);
277     if(methAnno!=null){
278     if (m.getReturnType().getClass().isInstance(Number.class)) {
279         Method targetMethod=clazz.getDeclaredMethod(methAnno.targetMethodName
                ), methAnno.targetMethodParam());
280         genValue =    Double.parseDouble(m.invoke(this).toString());
281         if(getMaxValue()!=null)
282         currentMaxVariance=  (getMaxValue().getClass().getDeclaredField(
                methAnno.min_maxField())).getDouble(getMaxValue());
283         if(getMinValue()!=null)
284             currentMinVariance=    (getMinValue().getClass().getDeclaredField(
                    methAnno.min_maxField())).getDouble(getMinValue());
285         if (config.getMutationRate() > Math.random()) {
286             double range;
287             range = currentMaxVariance-currentMinVariance;
288             genValue=currentMinVariance +  (int) (range * Math.random());
289         }
290         if(methAnno.targetMethodParam().isPrimitive()){
291             targetMethod.invoke(this, Number.class.getDeclaredMethod(
                    methAnno.targetMethodParam().getName()+"Value").invoke(
                    genValue));
292         }
293     }
294     }
295     }
296
297     } catch (SecurityException e) {
298         logger.error(e);
299     } catch (NoSuchMethodException e) {
300         logger.error(e);
301     } catch (IllegalArgumentException e) {
302         logger.error(e);
303     } catch (IllegalAccessException e) {
304         logger.error(e);
```

```
305     } catch (InvocationTargetException e) {
306         logger.error(e);
307     } catch (NoSuchFieldException e) {
308         logger.error(e);
309     }
310
311
312 }
313
314 public  T cloneGenome(Class<T> clazz) {
315 T  result = null;
316 ByteArrayOutputStream baos = new ByteArrayOutputStream();
317     try {
318      new ObjectOutputStream(baos).writeObject(this);
319      ByteArrayInputStream bais = new ByteArrayInputStream(baos.toByteArray())
         ;
320      result = clazz.cast(new ObjectInputStream(bais).readObject());
321     } catch (IOException e) {
322       e.printStackTrace();
323     } catch (ClassNotFoundException e) {
324       e.printStackTrace();
325     }
326
327 return result;
328 }
329
330 }
```

A.2 Vergleich von Gleichverteilung und Normalverteilung

Listing A.2: Vergleich von Gleichverteilung und Normalverteilung

```
1  public void testVerteilung(){
2  for (int j = 0; j < 6; j++) {
3  int minWert =120;
4  int maxWert=130;
5  int intervall=maxWert-minWert;
6  int gleichVerteilterWert=(int)(minWert+(intervall*Math.random()));
7  int normalVerteilterWert=gleichVerteilterWert;
8  double mutationDegree=0.8;
9  intervall=(int)(intervall*mutationDegree);
10 Random random=new Random();
11 for (int i = 0; i < 6; i++) {
12     random=new Random();
13     //random.nextDouble() liefert eine gleichverteilte Zufalls-Zahl zwischen
           0.0 und 1.0 (Wird von Math.random() benutzt)
14     //Quelle : http://java.sun.com/j2se/1.4.2/docs/api/java/util/Random.html#
           nextDouble()
15     //Quelle : http://java.sun.com/j2se/1.4.2/docs/api/java/lang/Math.html#
           random()
16     //random.nextGaussian() liefert eine normalverteilte Zufalls-Zahl ()
17     //Quelle : http://java.sun.com/j2se/1.4.2/docs/api/java/util/Random.html#
           nextGaussian()
18     gleichVerteilterWert=gleichVerteilterWert + (int) ((intervall / 2) * (
           random.nextDouble() - 0.5));
19     normalVerteilterWert=normalVerteilterWert + (int)((intervall / 2) *
           random.nextGaussian());
20     System.out.println(("Gleichverteilt_"+i+" ")+ gleichVerteilterWert);
21     System.out.println(("Normalverteilt_"+i+" ")+ normalVerteilterWert);
22     //Um die Benutzereingaben zu simulieren, wird maximal 10 sek gewartet.
23     int sleepTime=(int) (random.nextInt(10000));
24     Thread.sleep(sleepTime);
25
26     }
27     }
28 }
```

Iteration 1, Gen-Wert 121

Gleichverteilt	**Normalverteilt**
122	117
122	118
123	124
123	126
122	121
122	120

Iteration 2, Gen-Wert 123

Gleichverteilt	**Normalverteilt**
122	126
122	130
121	133
122	134
123	128
123	131

Iteration 3, Gen-Wert 124

Gleichverteilt	**Normalverteilt**
124	127
124	127
123	124
123	121
122	118
123	121

Iteration 4, Gen-Wert 122

Gleichverteilt	**Normalverteilt**
122	118
121	117
121	119
121	119
120	113
120	113

A.3 Überprüfung der fachlichen Anforderungen

Universität Hamburg

Department Wirtschaftswissenschaften
Arbeitsbereich Marketing und Innovation
Prof. Dr. Thorsten Teichert

UHH · Marketing und Innovation · 3079 · Von-Melle-Park 5 · D-20146 Hamburg

Tel. 040-428 38 4643
Fax 040-428 38 5250
E-Mail: ami@econ.uni-hamburg.de

Hamburg, den 10.03.2008

Funktionsbestätigung EVO-Online

Herr Ulas hat eine voll funktionsfähige Software zur interaktiven Gestaltung von Produktdesign auf Basis von evolutionären Algorithmen entwickelt. Die Software, welche unter Firefox im Internet und somit ortsungebunden abrufbar ist, ermöglicht durch modularen Aufbau und weitgehend freier Parametrisierungswahl ein breites Spektrum von Anwendungen, welche für den Marketingkontext von Relevanz sind.

Die Software besteht aus zwei voneinander getrennten Ebenen: die Administrationsebene, in der neuen Befragung angelegt und bestehende Befragungen modifiziert werden können, sowie die Befragungsebene, in welcher die eigentlichen Befragungen stattfinden.

Die Administrationsebene ermöglicht mit wenigen, menügesteuerten Schritten die Konzipierung von evolutionären Studien für den Praxisalltag. Nach freier Eingabe von Erhebungsname und Beschreibung können drei verschiedene Konfigurationen spezifiziert werden, die bei freier Wahl von Mutationsrate und –Radius, Generationenanzahl, Populationsgröße sowie den beiden wesentlichen EA-parametern eine große Bandbreite von evolutionären Erhebungen ermöglichen. Durch Vergabe von Gruppennamen und Passwörtern kann ein selektiver Zugriff von Befragten auf parallel laufende Erhebungen sichergestellt werden. Somit ermöglicht die Software die simultane Durchführung von verschiedenen parallelen Studien.

Kernstück der Befragungsgestaltung bildet die Abbildung der evolutionären konfigurierten Designalternativen. Hier ist Herrn Ulas eine besonders elegante Lösung gelungen, welche eine Integration von quasi frei wählbaren Designkomponenten ermöglicht. Grundidee ist, dass ein Produktdesign zerlegt wird in verschiedene Komponenten, welche jeweils separat als Bilddatei eingelesen und von Farbe sowie Position her benutzerdefiniert frei variierbar sind. Die beispielhaft generierten Produkte aus für T-Shirts wie auch Flaschen belegen das breite Einsatzpotenzial des gewählten Ansatzes.

Die prototypische Befragung als solche ist bestechend klar aufgebaut und beinhaltet beispielhaft die typischen Anforderungen von realen Markterhebungen. So besteht sie aus einer einleitenden kurzen Abfrage von soziodemographischen Merkmalen, mit welcher die Zielgruppenadäquanz des Befragten sichergestellt werden kann, sowie der anschließenden Hauptbefragung. In mehreren Evolution stufen werden jeweils in die generierten Produktalternativen zur Auswahl vorgelegt, mittels einfacher Checkmarks erfolgt die Auswahl und die automatische Überleitung in die nächste Evolutionsstufe.

Herr Ulas ist es auf vorbildlicher Weise gelungen, drei verschiedene Sichtweisen in seiner Software zu integrieren: die anwendungsorientierte Sicht des Marketings, die Methodensicht der evolutionären Algorithmen sowie die datenbanktechnische Gestaltung im Hintergrund der Internetsoftware. Lediglich wäre eine verbesserte Instruktion auf Administrationsebene durch ein umfangreicher gestaltet es Handbuch oder aber auch Online-Hilfetasten wünschenswert gewesen.

In Gesamtbeurteilung ist festzuhalten, dass die Arbeit von Herrn Ulas sämtliche Anforderungen, die aus Nutzersicht für einen derartigen Software-Prototyp zu stellen sind, vollständig und zu vollster Zufriedenheit erfüllt.

Hamburg den 10.3.2008

Prof. Dr. Thorsten Teichert

A.4 Inhalt der CD-ROM

Dieser Arbeit ist eine CD-ROM beigefügt. Sie beinhaltet eine PDF-Version dieses Dokuments, das Handbuch zur EvoMarketOnline-Anwendung, den gesamten Programm-Quellcode der EvoMarketOnline-Anwendung, eine vorkompilierte Version der EvoMarketOnline-Anwendung, den Datenbank-Server und den Applikation-Server.
Die CD hat folgenden Verzeichnisstruktur:

- Diplomarbeit_Bülent_Ulas.pdf
- evo_admin_handbuch.pdf
- **source**

  der Programm-Quellcode der EvoMarketOnline-Anwendung
- **sql**

  evo_db_create.sql

  evo_db_insert_proff.sql
- **Web Applikation**

  evoOnlineV2
- **Software**

  apache-tomcat-6.0.14.zip

  mysql-5.1.23-rc-win32.zip

Versicherung über die Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 12. März 2008

Ort, Datum

Unterschrift