OO-Entwurf von Evolutionären Algorithmen

Stefan Walter

Lehrstuhl XI

10.04.2003

Gliederung

- Systematische Herangehensweise
  - Grundprinzipien und Beispiele
- TEA
- DREAM
- JEO – Java Evolving Objects
- Entwurf eines EA mit Mitteln der OOP

Gliederung

- Systematische Herangehensweise
  - Grundprinzipien und Beispiele
- TEA
- DREAM
- JEO – Java Evolving Objects
- Entwurf eines EA mit Mitteln der OOP

Systematische Herangehensweise

- Allgemeiner OO – Entwurf
  - Anwendungsfälle („use cases“) beschreiben
  - Strukturierung des Problems
  - Isolieren der Elemente, die als Klassen in Frage kommen
  - Beschreibung der Kommunikation der Klassen
  - Beschreibung der Hierarchien (Vererbung, Aggregation)

Systematische Herangehensweise

- Mögliche Strukturierung von EA
  - „passiv“
    - Population
      - Chromosomen
    - Insel
      - Umgebungen (environments)
  - „aktiv“
    - Initiatoren
    - Brüter (breeders)
    - Selektoren (selectors)
    - Bewerter (assessors)
    - Übersiedler (migrators)
    - Abbrecher (terminators)

Inselmodell

- Zur parallelen Ausführung des gleichen Algorithmus auf mehreren Maschinen
- Austausch von Informationen durch Migration
  - Individuen werden zwischen den Inseln ausgetauscht
  - Auswahl durch spezielle Operatoren („migrators“)
- Beeinflussung durch Parameter
  - Häufigkeit der Migration
  - Anzahl der Ausgetauschten Individuen
  - Auswahl der Inseln mit denen kommuniziert wird (Nachbarschaft)
  - Häufiger Austausch vieler Individuen -> panmikitsche Population

Das Panmikitische Modell

- Quasi eine große Population
- Parallelisierung von Mutation und Funktionsauswertung, da beschränkt auf Individuen
- Individuen werden auf verschiedene Prozessoren verteilt
- Crossover und Selektion wird komplizierter

Gliederung

- Systematische Herangehensweise
  - Grundprinzipien und Beispiele
- TEA
- DREAM
- JEO – Java Evolving Objects
- Entwurf eines EA mit Mitteln der OOP

TEA – Toolbox for Evolutionary Algorithms

- Ergebnis einer früheren PG am Lehrstuhl XI
- Ziel: Ermöglichung einer „natürlichen“ Repräsentation des Problems ohne Festlegung auf Standardrepräsentationen:
  - Binäre Kodierung (Genetische Algorithmen, GA)
  - Reele Vektoren (Evolutionsstrategien, ES)
  - Bäume (Genetische Programmierung, GP)

TEA – Toolbox for Evolutionary Algorithms

- Struktur (aktiver Teil)
  - teaPopulation (Population)
  - teaIndividual (Individuum)
    - teaIIntVector
    - teaIMultiChromo
    - teaISimple
  - Chromosome
    - teaCPermutation

TEA – Toolbox for Evolutionary Algorithms

- Struktur (aktiver Teil)
  - teaMutate (Mutationen)
    - Jeweils Unterklassen für die entsprechende Datentypen
  - teaRecombine
    - Ebenfalls Unterklassen für die entsprechende Datentypen
  - teaReplace (Ersetzungen)

Gliederung

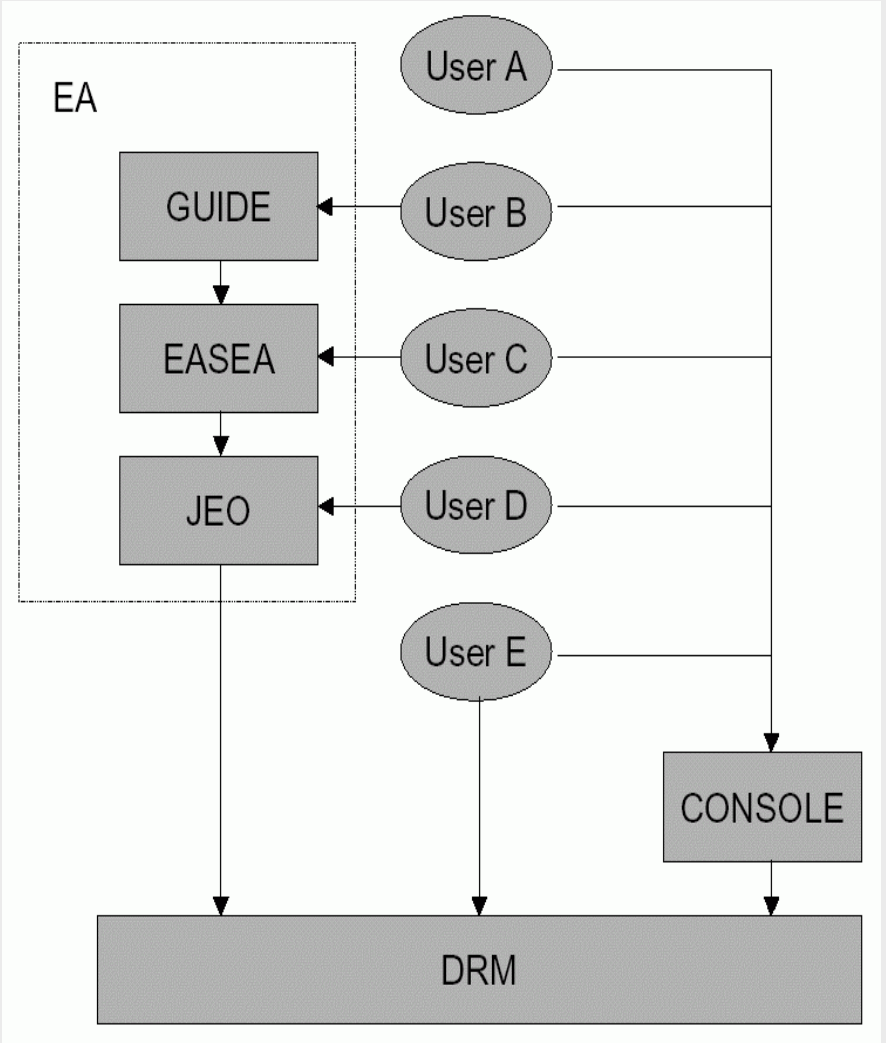
- Systematische Herangehensweise
  - Grundprinzipien und Beispiele
- TEA
- DREAM
- JEO – Java Evolving Objects
- Entwurf eines EA mit Mitteln der OOP

DREAM

- „Distributed Resource Evolutionary Algorithm Machine“
- Basis für verteilte evolutionäre Algorithmen
- Kommunkation über das Internet
- „CPU sharing“ für komplexe Algorithmen
  - Skalierbare Auslegung
- Verschiedene Einstiegspunkte für Benutzer

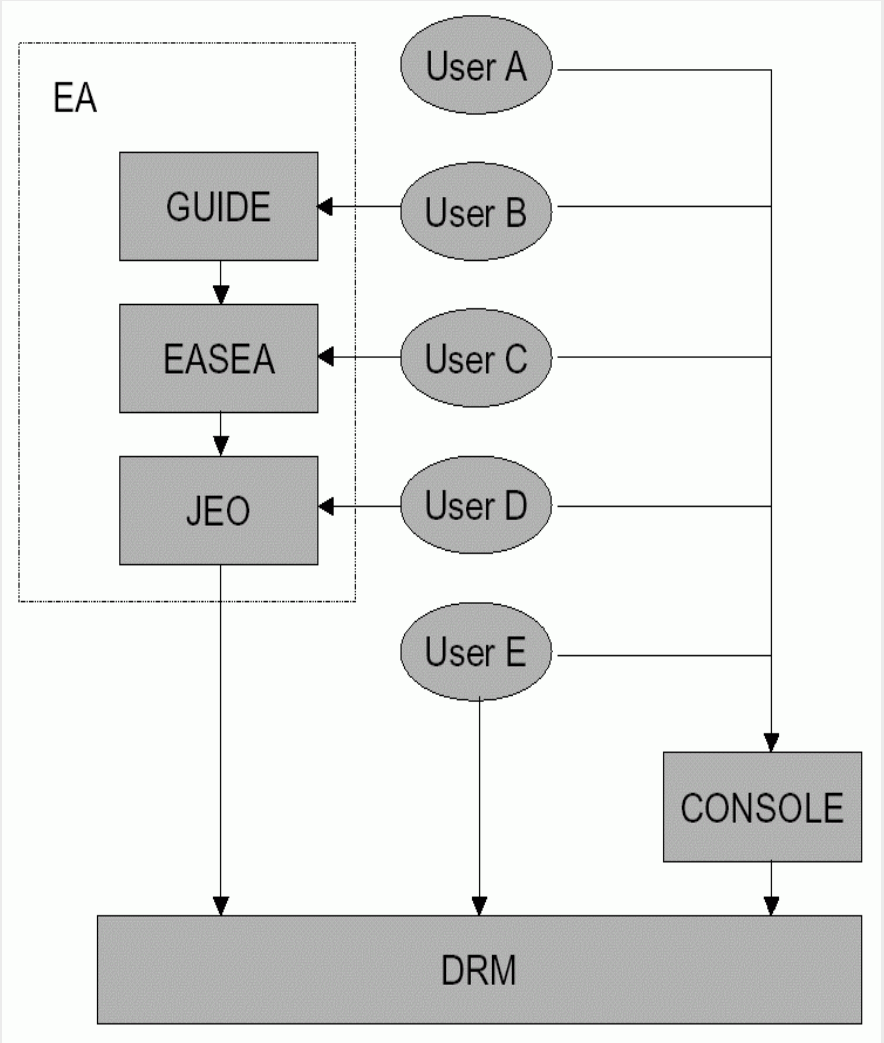
DREAM

- Struktur der Benutzung:



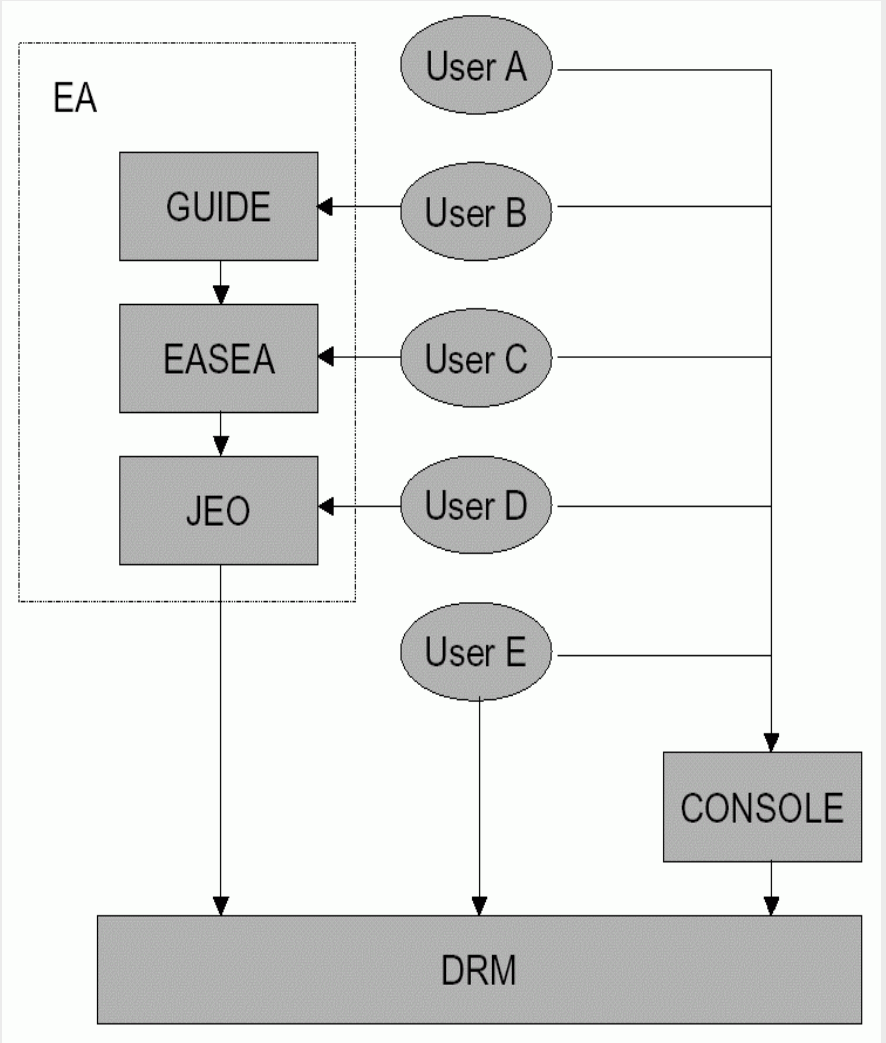
DREAM

- **Benutzer A** möchte
  - Rechenzeit zur Verfügung stellen
  - Experimente anderer überwachen
- Benutzer A benutzt die grafische Konsole



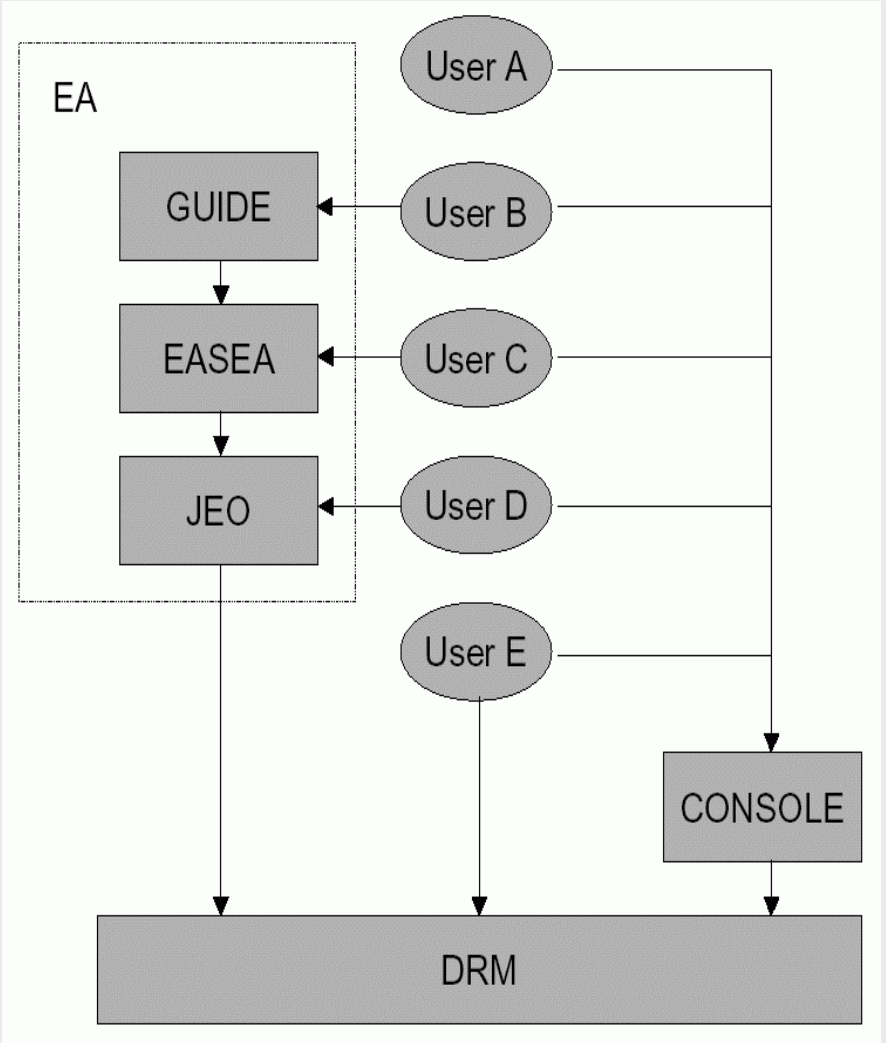
DREAM

- **Benutzer B** möchte
  - Algorithmen entwerfen
  - Nicht auf Textbasis programmieren
- Benutzer B interagiert mit GUIDE Schicht
  - Beeinflussung der evolution engine durch Parameter
  - Dadurch beliebige EA möglich
  - Problemspezifische Einstellungen mit Java-ähnlicher Hochsprache



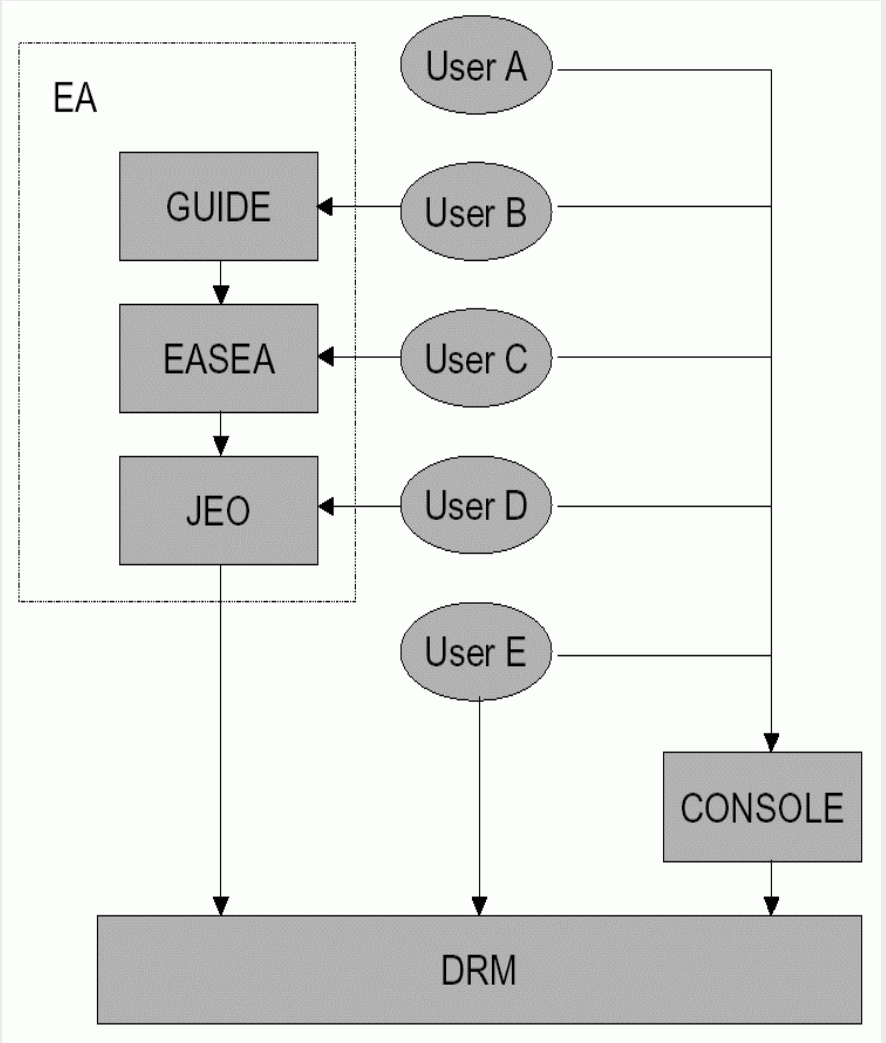
DREAM

- **Benutzer C** möchte
  - Das System durch EASEA programmieren
  - Hochsprachliche EA Entwicklungsumgebung
  - Produziert Java code, benutzt die JEO Bibliothek
  - Dateien können direkt vom Java compiler bearbeitet werden



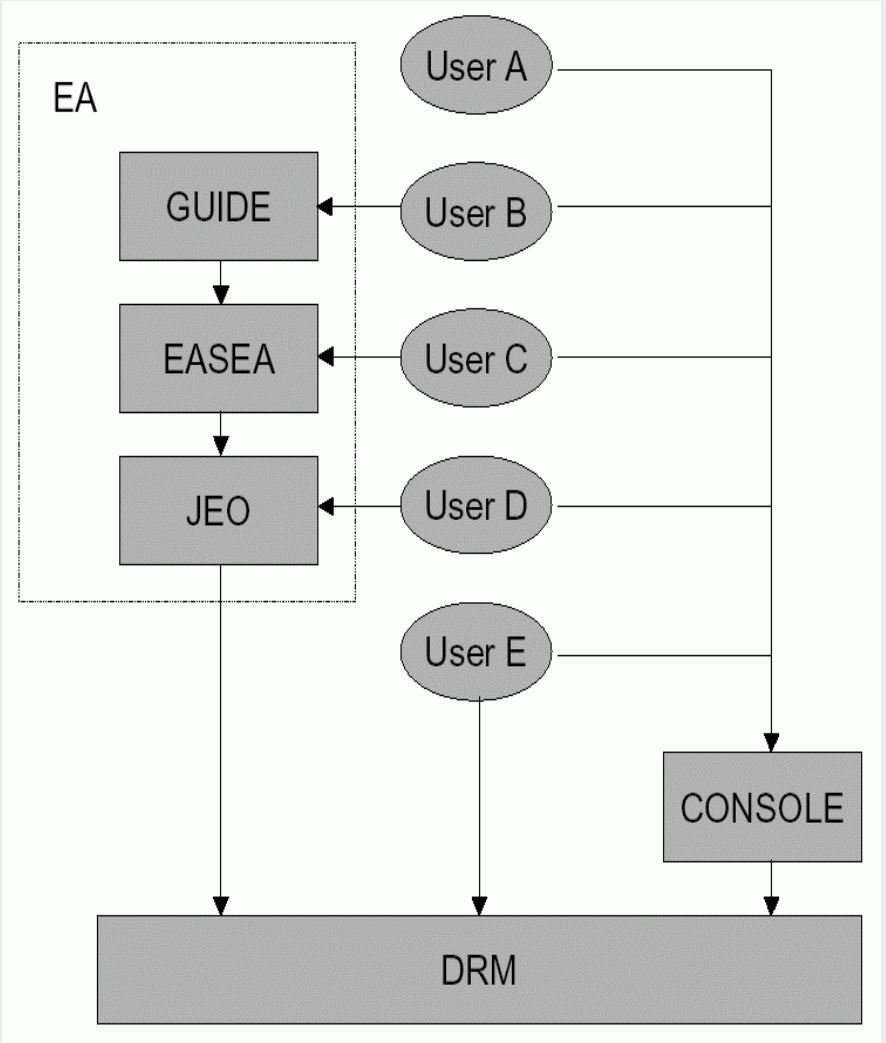
DREAM

- **Benutzer D**
  - Programmiert direkt in Java
  - Benutzt die JEO Bibliothek
    - Objekte müssen entsprechende Interfaces aus JEO implementieren



DREAM

- **Benutzer E**
  - Ist Experte und programmiert direkt mit Hilfe DRM API
  - Benutzt den Kern des Systems, die DRM (Distributed Resource Machine)



Gliederung

- Systematische Herangehensweise
  - Grundprinzipien und Beispiele
- TEA
- DREAM
- JEO – Java Evolving Objects
- Entwurf eines EA mit Mitteln der OOP

JEO – Java Evolving Objects

- Ist eine Java Bibliothek für die Durchführung evolutionärer Experimente
- Flexibel ➡ für fast jedes Experiment geeignet
- Basiert auf dem Inselmodell ➡ ermöglicht verteilte Ausführung

JEO – Java Evolving Objects

- Struktur:
  - Folgt den Richtlinien der Objektorientierten Programmierung
    - Objekt der gleichen Klasse können ausgetauscht werden ➡ verändert nur die Spezifikation des Experiments
    - JEO ist benutzbar wie ein LEGO-Kasten
    - Erweiterbarkeit durch Implementierung von Interfaces
    - Mobil und portabel ➡ Implementierung von `java.io.Serializable`

JEO – Java Evolving Objects

- InfoHabitants
  - Individuen in DREAM
  - Besteht aus Genom (Problemlösung) und Fitness
  - Braucht Resourcen, um auf einer Insel zu überleben

JEO – Java Evolving Objects

- Genome
  - Evolvierbarer Teil des Individuums
  - Set aus Genen = Problemlösung
  - Operationen
    - Initialisierung (durch Inter Objekt)
    - Modifikation (durch Operator Objekt)
  - Es stehen viele verschiedene Genome zur Verfügung

JEO – Java Evolving Objects

- Fitness
  - Deutet auf Güte der Lösung hin
  - Einzige Bedingung an die Fitness: Vergleichbarkeit
    - Erreicht durch erben von Interface `java.lang.Comparable`
    - Jedes Objekt das von o.g. Interface erbt, kann Fitness sein

JEO – Java Evolving Objects

- Breeders
  - Wendet Variationsoperatoren an Population an
  - Erzeugt Nachkommengeneration
  - Benutzt eine Vielzahl von Variationsoperatoren
  - Kontrolliert die Anwendung der Variationsoperatoren, bis Population spezifizierte Größe erreicht hat

JEO – Java Evolving Objects

- Selectors
  - Auswahl eines InfoHabitants aus der Population
  - Für verschiedene Operationen benutzt
    - Zeugung: Selektion von Eltern für Produktion von Nachkommen
    - Migration: Emigrator sucht InfoHabitant zum Auswandern, ImMigrator sucht lokalen InfoHabitant, der durch Einwanderer ersetzt wird
    - Reward: Rewarder suchen InfoHabitants, die belohnt werden sollen
  - Verschieden Versionen für verschiedene Strategien

JEO – Java Evolving Objects

- Assessors (Bewerter, Schätzer)
  - Evaluierung und Ersetzung
    - Sollte immer zusammen (nacheinander) ausgeführt werden
    - Zusammengefasst in Assessor
  - Ausführung nach Veränderungen

JEO – Java Evolving Objects

- Migrators
  - Dienen dem Austausch von Lösungen zwischen den Inseln
  - Erhöhen die Diversizität der Inselbevölkerung
  - Kontrolle über Häufigkeit und Umfang des Austausches

JEO – Java Evolving Objects

- Terminators
  - Überprüfen Abbruchbedingungen und beenden evntl. die Evolution
  - Abbruchbedingungen können einfach (Zähler) oder komplex sein (Qualität der Lösung)
  - Außerdem statistische Funktionen

Gliederung

- Systematische Herangehensweise
  - Grundprinzipien und Beispiele
- TEA
- DREAM
- JEO – Java Evolving Objects
- Entwurf eines EA mit Mitteln der OOP

Entwurf von EA mittels OOP (Fazit)

- Umsetzung der Rollen eines EA fast zwingend
  - Übersetzung in entsprechende Klassen
- Beachtung objektorientierter Prinzipien wie bei anderen Programmentwürfen
  - Design der Schnittstellen
  - Kapselung
  - Vollständigkeit
  - Geheimhaltung
  - Etc.

Entwurf von EA mittels OOP (Fazit)

- Vorteile durch OO-Ansatz
  - Austauschbarkeit von Objekten
    - Funktionale Elemente von verschiedenen EA
  - Delegation
    - Änderung des Algorithmus zur Laufzeit
  - Vererbung
    - Spezialisierungen auf Basis einer allgemeinen Klasse
- Ziel:
  - Kleinsten gemeinsamen Nenner finden für ein flexibles Klassenmodell
  - Darauf aufbauend spezielle Ansätze

ENDE