The Evolutionary Approach to Semantics-Driven CBD Automation

Yu Jia
Institute of Software, Chinese Academy of Science
Beijing 100080, China
jia_yu@263.net
Two Problems in CBD Methodology

CBD Process:
- Multi-phrases:
  - Retrieve, qualification, adaptation, composition, deployment
- Different views to each phrase. Various techniques in each phase.

Problem 1: CBD paradigm lacks a commonly accepted principle to harmonize its engineering activities.

Component Semantics:
- IDLs are originally targeted for composition and communication.
- Documents, test cases... to be used as semantic carriers.

Problem 2: The component semantics is presented informal and independent from the component itself.
A Interesting Question

- EAs are search and optimization processes founded on self-organized individuals.
- the CBD is a software building procedure based on self-contained and highly independent components.

Is EA theory appropriate for CBD paradigm?
CBD vs. EA: The Corresponding Properties and Behaviors

- Self-organization
- Self-adaptation
- Self-learning
- Survival of the fittest
- Process
- Multi-solution and population search
- Statistics and random
- Intrinsic Parallelism
- Global optimization
CBD/EA theory

• The component is regarded as the evolvable individuals. The CBD process as the evolutionary procedure. So the algorithms in EA can be used as the automating tools in CBD.

• The Feature Modeling method is applied to describe the content and structure of component semantics.
  
  **Genotype:** The Feature Description Logics is used to code the component
  
  **Phenotype:** The meaning of Feature in a specific domain

• The user req. in Feature Space is used to calculate the fitness.

• All component-process activities can be categorized as 3 Operations: reproduction, crossover, and mutation.

• The object is to find or produce the nearly best solution in the individual space: The target component-based system.
The Core of CBD/EA Approach

- A component model:
  Evolutionary component
- A engineering method:
  Feature Modelling
- A formal language:
  Feature Description Language
- A automation process:
  The evolutionary algorithms for CBD
The Evolvable Component Model

- Evolvable components are software entities implemented in CBD framework techniques and encoded in EA theory.
- Coming from the 3C model:
  
  \[ \text{component} = (\text{CONCEPT}, \text{CONTENT}, \text{CONTEXT}) \]

  \[ \text{CONCEPT} = (\text{SEM\_INTERFACE}, \text{SYN\_INTERFACE}) \]

  \[ \text{CONTEXT} = (\text{SEM\_CONFIG}, \text{SYN\_CONFIG}, \text{DOMAIN}) \]

- Considering only the semantic parts:

  \[ \text{semantics} = (\text{DOMAIN}, \text{SEM\_INTERFACE}, \text{SEM\_CONFIG}) \]

- Implemented in Feature Space:

  \[ \text{semantics} = (\Omega_{\text{dom}}, \Omega_{\text{def}}, \Omega_{\text{con}}) \]
Feature Modeling

- “Feature” refers to concepts in a domain, mostly the services terms in perspective of user.
- A engineering methods in Separation of Concerns. The ontology theory in AI is its base.
- In CBD/EA, it used as:
  - A domain modelling engineering method;
  - A component semantics describing unit;
  - A unit to define user requirements;
  - The tools for analyzing or predicting the attributes of compositional system;
  - A base to design component evolution procedure and algorithms.

Book: K. Czarnecki, Generative Programming: Methods, Techniques, and Applications
Feature Modeling Techniques in CBD/EA

- The feature space:
  - Domain Space, Definition Space, Context Space, and Requirement Space
- The feature type:
  - Mandatory, optional, alternative, and OR feature
- The visual describing language: the Feature tree.
- A formal describing language: the feature description language.
- A set of feature model analyzing and verifying methods.
- Data structure and computing algorithms for feature space.
- A feature-oriented engineering framework:
  - The life cycle, A model specification, A feature dictionary
  Stakeholder’s task,…
Feature tree: an example

Feature Tree for Domain Space

Feature Tree for Definition Space

Feature Tree for Context Space
Feature Description Logics (FDL)

- Description Logics (DLs) are knowledge representation languages for expressing knowledge about concepts and concept hierarchies.
- In DLs the domain of interest is modeled by means of individuals, concepts, roles and knowledge base exactly corresponding to the feature items, features, feature relationships and Feature Space respectively.
- The role of FDL in CBD/EA:
  - A formal description language for component semantics;
  - A knowledge-based on the reuse method;
  - A intelligent tool to support CBD automation;
  - A genetic coding for evolutionary component
### The Syntax and semantics of FDL concept and role constructs

<table>
<thead>
<tr>
<th>Features (Concepts) $C_\varphi$</th>
<th>Syntax $\rho$</th>
<th>Semantics $\varphi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>atomic feature $\varphi$</td>
<td>$A_\rho$</td>
<td>$A^I \subseteq \Delta^I_\varphi$</td>
</tr>
<tr>
<td>universal feature $\varphi$</td>
<td>$T_\rho$</td>
<td>$\Delta^I_\varphi$</td>
</tr>
<tr>
<td>negation $\varphi$</td>
<td>$\neg C_\rho$</td>
<td>$\Delta^I \setminus C^I_\varphi$</td>
</tr>
<tr>
<td>conjunction $\varphi$</td>
<td>$C_1 \cup C_2_\rho$</td>
<td>$C^I_1 \cup C^I_2_\varphi$</td>
</tr>
<tr>
<td>alternation $\varphi$</td>
<td>$C_1 \lor C_2_\rho$</td>
<td>$(C^I_1 \cap \neg C^I_2) \cup (\neg C^I_1 \cap C^I_2)_\varphi$</td>
</tr>
<tr>
<td>universal role quantification $\varphi$</td>
<td>$\forall R.C_\rho$</td>
<td>${ o \mid \forall o' : (o, o') \in R^I \rightarrow o' \in C^I }_\varphi$</td>
</tr>
<tr>
<td>quantified number restriction $\varphi$</td>
<td>$(\leq n P.C)_\rho$</td>
<td>${ o \mid #{o' \mid (o, o') \in P^I \land o' \in C^I } \leq n }_\varphi$</td>
</tr>
<tr>
<td>collection of individuals $\varphi$</td>
<td>${a_1, \ldots, a_2}_\rho$</td>
<td>${a^I_1, \ldots, a^I_2}_\varphi$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Feature Relations (Roles) $R_\varphi$</th>
<th>Syntax $\rho$</th>
<th>Semantics $\varphi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>atomic feature relations $\varphi$</td>
<td>$P_\rho$</td>
<td>$P^I \subseteq \Delta^I \times \Delta^I_\varphi$</td>
</tr>
<tr>
<td>disjunction $\varphi$</td>
<td>$R_1 \cup R_2_\rho$</td>
<td>$R^I_1 \cup R^I_2_\varphi$</td>
</tr>
<tr>
<td>reverse $\varphi$</td>
<td>$R^-_\rho$</td>
<td>${(o, o') \in \Delta^I \times \Delta^I \mid (o', o) \in R^I }_\varphi$</td>
</tr>
<tr>
<td>concatenation $\varphi$</td>
<td>$R_1 \circ R_2_\rho$</td>
<td>$R^I_1 \circ R^I_2_\varphi$</td>
</tr>
<tr>
<td>reflect transitive closure $\varphi$</td>
<td>$R^*_\rho$</td>
<td>$(R^I)^*_\varphi$</td>
</tr>
</tbody>
</table>
The Component Semantics in FDL

<table>
<thead>
<tr>
<th>Notation Names</th>
<th>Expressions in FDL</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature Item</td>
<td>(C(a), D(b), D(b))</td>
<td>Person (zhong), Child(yinyin)</td>
</tr>
<tr>
<td>Item Relation</td>
<td>(R(a, b))</td>
<td>hasChild(zhong,yinyin)</td>
</tr>
<tr>
<td>Composite Item</td>
<td>(C(a), D(b), D(b))</td>
<td>Person (zhong), Child(yinyin), Child(jiejin)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hasChild(zhong,yinyin)</td>
</tr>
<tr>
<td>Control Item</td>
<td>(C(a), \forall R \downarrow (a))</td>
<td>Person (zhong)</td>
</tr>
<tr>
<td>Basic Item</td>
<td>(C(a), \forall R \downarrow (a))</td>
<td>University(tsinghua)</td>
</tr>
<tr>
<td>Shared Item</td>
<td>(C(a), C(a), D(b), D(b), \exists R \downarrow (a), \exists R \downarrow (b))</td>
<td>hasStatus(yinjin, 4th_School), hasStatus(xinjin, 4th_School)</td>
</tr>
<tr>
<td>Alternative Relation</td>
<td>(C(a), D(b), D(b))</td>
<td>Child(jiejin), University(tsinghua), Child(jiejin), University(beiying), Child(jiejin), University(beiing), University(beiing)</td>
</tr>
<tr>
<td>Mandatory Relation</td>
<td>(C(a), D(b), R(a, b))</td>
<td>Person (zhong), Child(yinyin), hasChild(zhong,yinyin)</td>
</tr>
<tr>
<td>Optional Relation</td>
<td>(C(a), D(b), (\exists R \downarrow (b)))</td>
<td>Child(yinyin); School(eveing), Child(yinyin)</td>
</tr>
</tbody>
</table>

**Domain Space in FDL**

- Feature: \(A, B, C, D, E, F\)
- Feature Relation: \(P, R, Q\)
- Composite Feature: \(C \subseteq E\), \(C \subseteq D \cup (E \cap \text{age})\)
- Event Feature: \(C \subseteq R\)
- Atomic Feature: \(A \subseteq R\)
- Not Feature: \(C \subseteq \neg D\)
- Shared Feature: \(C \subseteq R \cup (B \cup E)\)
- Aggregative Relation: \(C \subseteq D \cup (E \cap D)\)
- Instantial Relation: \(C \subseteq E \cup D\)
- Crossover Relation: \(C \subseteq D \cup (E \cap D)\)
- Alternative Relation: \(C \subseteq D \cup (\neg E)\)
- OR-Relation: \(C \subseteq D \cup (E \cup \neg D)\)
- Mandatory Relation: \(C \subseteq D \cup (E \cap \neg D)\)
- Optional Relation: \(C \subseteq (D \cap E)\)
- Non-restrict Relation: \(C \subseteq (D \cap \neg E)\)
- Composite Relation: \(R \subseteq Q \cup D\)

**Definition Space in FDL**

<table>
<thead>
<tr>
<th>Notation Names</th>
<th>Expressions in FDL</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature Parameter</td>
<td>(R(C(a)))</td>
<td>hasStatus(zhong,yinyin)</td>
</tr>
<tr>
<td>Parameter Relation</td>
<td>(R(a))</td>
<td>hasChild(zhong,yinyin)</td>
</tr>
<tr>
<td>Composite Parameter</td>
<td>(R(C(a)), D(b))</td>
<td>hasStatus(zhong,yinyin), hasStatus(child,yinjin)</td>
</tr>
<tr>
<td>Independent Parameter</td>
<td>(R(C(a)))</td>
<td>Person (zhong)</td>
</tr>
<tr>
<td>Basic Parameter</td>
<td>(C(a), \forall R \downarrow (a))</td>
<td>University(tsinghua)</td>
</tr>
<tr>
<td>Shared Parameter</td>
<td>(R(C(a)), D(b))</td>
<td>hasStatus(zhong,yinyin), hasStatus(child,bubin)</td>
</tr>
<tr>
<td>Mandatory Relation</td>
<td>(C(a), D(b))</td>
<td>Person (zhong), Child(yinyin), hasChild(zhong,yinyin)</td>
</tr>
</tbody>
</table>

**Context Space in FDL**

28th Euromicro conference CBSE Sept. 6, 2002
Institute of software, Chinese Academy of Science
The Power of FDL

The Basic FDL reasoning services:

- Feature Satisfiability
- Subsumption
- Consistency
- Instance Checking

CBD problems can be solved by FDL reasoning:

- Component categorization and retrieval
- Feature interaction
- Semantics mismatch
- Component assembly attributes prediction
The Computational Model of CBD/EA

- Representation Scheme
- Fitness Measure
- Selection Mechanism
- Genetic Operators
- Terminal Criteria
The Computational Model of CBD/EA: Representation Scheme

Coding component in FDL


(2) Problem: $\Sigma = \{C1(a1), C11(b1), C12(b2), C2(c1), C2(c2), P11(a1, b1), P12(a1, b2), p21(b1, c1), p22(b2, c2)\}$

(3) Component population: $\Sigma_1 = \{C(a1), C1(a1), C3(a2), C11(b1), C12(b2), P1(a1, a1), P2(a1, a2), P11(a1, b1), P12(a1, b2)\}; \Sigma_2 = \{C11(a1)\}; \Sigma_3 = \{C2(c1)\}; \Sigma_4 = \{C2(c2)\}; \Sigma_5 = \{C11(b1), C2(c1), p21(b1, c1)\}; \Sigma_6 = \{C1(a1), C11(b1), C12, P11(a1, b1), P12(a1, C12)\}$
The Computational Model of CBD/EA: Fitness Measure

(1) Knowledge-based measurement.
   To computer the similarity between problem and component knowledge base;

(2) Tree-based measurement.
   To computer the similarity between problem and component feature tree;
In CBD/EA, a selective pressure-driven scheme is suggested according to the knowledge-based fitness, where the population is sorted into two groups in order of largest similarity (LC) value and least different (LD) value respectively.
The Computational Model of CBD/EA: Genetic Operators

- **Reproduction:**
  - Select a component and copying an existing component code.

- **Crossover:**
  - Create one or more new component codes from two existing component codes.

- **Mutation**
  - Create a new component code from an existing component code by mutating a randomly chosen part.
The Life Cycle of Evolvable Components in CBD/EA

- Registration
  - Component Repositories
  - Retrieval
  - Initial Population
  - Parent Components

- Test in Practice
  - Competitive Component in Technique and Market
  - Selection
  - Genetic Operations

- Composition and Deployment
  - Running Component
  - Best-so-far Component
  - New Generated Components
  - Invalidation

- Evaluation Fitness
  - Correct Components
Future work: A Component-based Software Developing Platform

Feature Description Logic Reasoning Machine

- Feature code creator
- Candidate Component Implementation
- Candidate Component Semantics Description
- Requirement Specification
- Domain Architecture Standard
- Component composition tools
- Component generator
- Component-based software system
Cooperation is welcome
Thank you....

Yu Jia
Institute of Software, Chinese Academy of Science
Beijing 100080, China
jia_yu@263.net