XML based Modelling of Soft Computing Methods

Christian VEENHUIS and Mario KÖPPEN and Katrin FRANKE
Fraunhofer Institute for Production Systems and Design Technology IPK,
Department Pattern Recognition,
Pascalstr. 8-9,
D-10587 Berlin, Germany,
Phone: ++49(0)30 / 390 061 94, Telefax: ++49(0)30 / 391 75 17,
Email: veenhuis@ipk.fhg.de , mario.koeppen@ipk.fhg.de , katrin.franke@ipk.fhg.de
Definition of Soft Computing

Soft computing differs from conventional (hard) computing in that, unlike hard computing, it is tolerant of imprecision, uncertainty, partial truth, and approximation. In effect, the role model for soft computing is the human mind. The guiding principle of soft computing is: Exploit the tolerance for imprecision, uncertainty, partial truth, and approximation to achieve tractability, robustness and low solution cost.

L. Zadeh,
(father of Fuzzy Logic and founder of Soft Computing)
## Multitude of Soft Computing Methods (SCM)

<table>
<thead>
<tr>
<th>Evolutionary Algorithms</th>
<th>Genetic Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Artificial Immune Systems</td>
<td>Fuzzy Logic</td>
</tr>
<tr>
<td>Genetic Programming</td>
<td>Artificial Life Algorithms</td>
</tr>
<tr>
<td>Interactive GA</td>
<td>Cultural Algorithm</td>
</tr>
<tr>
<td>Fuzzy GA</td>
<td>Ant Colony</td>
</tr>
<tr>
<td>Nessy</td>
<td>Co-evolution</td>
</tr>
<tr>
<td>Genetic Hillclimbing</td>
<td></td>
</tr>
<tr>
<td>PBIL</td>
<td></td>
</tr>
<tr>
<td>Scouts</td>
<td></td>
</tr>
<tr>
<td>Artificial Chemistry</td>
<td></td>
</tr>
</tbody>
</table>

**Evolutionary Programming**

**Neural Networks**

**Evolutionary Strategy**

**Fuzzy GA**

**Interactive GA**

**PBIL**

**Scouts**

**Artificial Chemistry**
XML based Modelling of Soft Computing Methods

State of the Art

State in the scientific community:

• Growing number of Soft Computing Methods and algorithms → reinventing of known SCM

• No standardized programming language or modelling concept → lack of exchange and discussions

• Everyone uses its preferred programming language and operating system

• Reprogramming, redefinition, or translation of implemented or new published algorithms

Application of SCMs:

• Implementation of your own algorithms (→ time consuming, stand-alone solutions)

• Use one of the available libraries (→ black box approach, limited adaptation opportunities)
XML based Modelling of Soft Computing Methods

Modelling for Soft Computing Methods (SCM)

- To revival efforts in exchange and standardization
- To support the modelling, description and documentation of a SCM at once

**Required characteristics:**
- System-independent
- Separated from an explicit implementation
- Easy to understand and to learn
XML based Modelling of Soft Computing Methods

**Common Nature of Algorithms and Documents**

- **Content**: e.g. text and images
- **Structure**: Logical information
- **Layout**: e.g. typography

**Usage of documents to “visualize” algorithms**

**Definition:**
An algorithm is a rule based, determined method.

e.g. keywords, graphics

- **e.g. flow chart, listing, syntax highlighting**
- **e.g. variable, equation, loop, command, control structure**
XML based Modelling of Soft Computing Methods

Approaches for a modelling concept

- Hypertext Markup Language (HTML)
- Extensible Markup Language (XML)

Meta-language for the description of structured documents
XML based Modelling of Soft Computing Methods

Extensible Markup Language (XML)

http://www.w3.org/XML

• XML has to be **used easily within the Internet**.
• It has to be easy to implement programs that process XML documents.
• The number of optional features within XML has to be minimal, ideal case is Zero.
• XML-documents have to be **readable and understandable by human**.
• The XML-design has to be realized rapidly.
• The **design of XML has to be formal and exactly**.
• XML-documents have to be created easily.
• The shortage of XML-markups is of minimal importance.
XML based Modelling of Soft Computing Methods

Notation and basic concept of XML

- Document Type Definition
- Element Notation

```xml
<?xml version="1.0"?>
<!DOCTYPE PERSONS SYSTEM "persons.dtd">

<PERSONS>
  <PERSON>
    <FIRSTNAME> Jack </FIRSTNAME>
    <SURENAME> McDevitt </SURENAME>
  </PERSON>
</PERSONS>
```

http://www.w3.org/XML
First efforts using XML for Soft Computing Methods:

- **EAML (Evolutionary Algorithm Modelling Language)**
  Description of a wide variety of EAs by nested operators
  URL: http://vision.fhg.de/~veenhuis/EAML

- **NNML (Neural Network Markup Language)**
  Description of complete Neural Network models
  URL: http://www.nnml.alt.ru

- **CAML (Cellular Automata Modelling Language)**
  Description of CAs by hierarchically defining rule-sets
  URL: http://vision.fhg.de/~veenhuis/CAML
<<xml version="1.0"?>
<!DOCTYPE EAML SYSTEM "eaml.dtd">

<EAML standalone="true" project="bits">

<Code name = "objfunc"> <![CDATA[
// this is the objective-function
  double obj = ECPARAM(CBits,gene,size); // the worst fitness is the number of bits

  // Compare whole genome bitwise with desired target.
  // Every equal bit-pair causes a decrementation.
  // Thus the best fitness is 0 and the worst is the number of bits within genome.

  for (int i = 0 ; i < ECPARAM(CBits,gene,size) ; i++ )
    if (GETDATA( i ) == 1) obj -= 1;
  return obj;
]]> </Code>

A sample EA written in EAML

Objective functions must be written in the desired programming language
<Algorithm name = "CBits"  <!-- the main-algorithm (we use only one) -->
  size = "20"
  direction = "minimize"
  generations = "100"
  optimum = "0.0"
  elitistRate = "10%"
  operatorRate = "90%"
>
  <!-- reference to objective-function -->
  <objective> <Use ref="objfunc"/> </objective>

  <!-- use every bit alone -->
  <genome> <BitString size="30" group="1"/> </genome>

  <!-- fitness-proportional selection -->
  <selection> <RouletteWheel/> </selection>

  <!-- This is the operator-structure. -->
  <operator> <Group> <operators>  
    <!-- We create two parallel operators: -->
    <!-- a binary (crossover p=0.9) and -->
    <!-- an unary (mutation p=0.1). -->
    <!-- After creating offspring with the binary -->
    <!-- operator, we additionally mutate the -->
    <!-- offspring with p = 0.1. -->
    <Binary rate = "90%" succRate = "10%" >
      <OnePoint/>
    </Binary>
    < Unary rate="10%">
      <PointIncrement min="0" max="1" step="1"/>
    </ Unary>
  </operators> </Group> </operator>

  <!-- initialize the bit-strings randomly -->
  <initial> <RandomInitial min="0" max="1"/> </initial>

</Algorithm>
</EAML>

Genetic operators are nested to build the specific EA
A sample snippet written in NNML

```xml
<?xml version="1.0" standalone="no"?>
<!DOCTYPE neuralmodel SYSTEM "nnml.dtd">

<neuralmodel>
  <nn_structure>
    <layer id="1">
      <neuron id="1">
        <combination_function> ...further elements... </combination_function>
        <activation_function> ...further elements... </activation_function>
      </neuron>
      <neuron id="2">
        ...further elements...
      </neuron>
    </layer>
    <layer id="2">
      ...further elements...
    </layer>
    ...further elements...
  </nn_structure>
  <nn_activity>
    <nn_constraints>
      <nn_constraint>
        <nn_parameter>weight</nn_parameter>
        <nn_parameter_range min="-1" max="1" brackets="11"/>
      </nn_constraint>
      ...further elements...
    </nn_constraints>
  </nn_activity>
  <neural_network>...further elements...</neural_network>
</neuralmodel>
```

Any number of neurons

Any number of layers
XML based Modelling of Soft Computing Methods

A sample CA written in CAML

```xml
<?xml version="1.0"?>
<!DOCTYPE CA SYSTEM "caml.dtd">

<!-- Sierpinsky triangle (cell-values: 0 = dead, 1 = alive) -->

<CA project="sierpinsky" dimensions="1" min="0" max="1">
  <Rule>
    <Cond>
      <OR>
        <Pattern dimx="3" centerx="1" > 0 0 0 </Pattern>
        <Pattern dimx="3" centerx="1" > 1 1 1 </Pattern>
      </OR>
    </Cond>
    <Action> <Current> <Literal value="0"/> </Current> </Action>
    <Otherwise> <Current> <Literal value="1"/> </Current> </Otherwise>
  </Rule>
  ...further rule elements...
</CA>
```

Logical and arithmetical expressions can be defined hierarchically.

Any number of rules can be defined.
XML based Modelling of Soft Computing Methods

A XML based Modelling Language for a Soft Computing Method

- Is a Domain-Specific Language
- Represents a whole Software System Family
- A concrete Description („Program“ or Model) represents an individual Software System of the appropriate Domain

Example:
EAML represents the domain of Evolutionary Algorithms. An EAML document models one individual EA of this domain.
Domain Engineering

- **Domain Analysis**
  Define the domain and build the **Domain Model**.

- **Domain Model**
  Contains all properties of the domain (e.g. attributes, parameters, operators, constraints, ...).

- **Domain Design**
  Create a very general algorithm of the domain.
  This algorithm should cover all possibilities of the defined domain.

- **Domain Implementation**
  Develop software-generators, interpreters, reusable components, libraries, aso.
XML based Modelling of Soft Computing Methods

Example: A possible EAML

• **Domain Analysis**
  - Encoding of solutions
  - Initialising functions and states
  - Objective functions and fitness functions
  - Termination criteria
  - Selection methods
  - Genetic operators
  - Probabilities (operators, selection methods,...)
  - Replacement schemes
  - Populations of solutions (individuals)
  - Constraints (gene value range, thresholds,...)

• **Domain Model**
  Specify the properties hierarchically
XML based Modelling of Soft Computing Methods

Example: A possible EAML

- **Domain Design**
  - A very general algorithm for the **Domain Model** was designed by using OOP methods.
  - The domain-methods are realized by using abstract classes for polymorphic interfaces.
  - Domain-components are instances of classes which are derived from the abstract classes.
  - The general algorithm gets combinations of these instances to specify an individual EA.

- **Domain Implementation**
  A software-generator was developed

A subset of the designed EA domain
XML based Modelling of Soft Computing Methods

OOP Model → Document Type Definition (DTD)

Every abstract class for a domain-method-interface gets its own XML element

Every component-class (derived from the abstract class) gets its own XML element

Every parameter of a domain-method is a class attribute. Every class attribute is realized as attribute of the XML element
XML based Modelling of Soft Computing Methods

Usage of the Specified Modelling Language

• Every domain-component is a XML element
• Elements can be considered as „building blocks“
• Hierarchical order describes a model
• XML document represents a single software system
• DTD represents the software system family

```xml
<?xml version="1.0"?>
<!DOCTYPE EA SYSTEM "EA.dtd">

<EA>
  <Genome>
    ...a nested element...
  </Genome>

  <Selection>
    <Tournament size = "10" />
  </Selection>

  <Operators>
    ...further nested elements...
  </Operators>

  ...further nested elements...
</EA>
```
XML based Modelling of Soft Computing Methods

How to create XML documents?

Creation

• With any XML editor
• With any usual text-editor (ASCII)
• Generated by other software systems

Automated Generation of Texture Filters by means of Evolutionary Algorithms

XML

document

DTD
XML based Modelling of Soft Computing Methods

How to use XML documents?

Usage (XML documents are used with the products of the Domain Implementation phase)

- Transformations by using XSL(T) style-sheets
- With a software-generator or compiler
- With an interpreter
XML based Modelling of Soft Computing Methods

Conclusions

- Soft Computing Methods can be specified in a hierarchical order
- Domain Engineering methods creates hierarchical Domain Models
- Hierarchical Domain Models can be transformed into Document Type Definitions
- XML documents for this DTD contain complete models of the SCM
- Such models are configurable by using attributes of the XML elements

Exchange of SCM models between software systems is possible by using XML and DTDs as modelling concept and „interface“
Discuss and exchange your algorithms!

The building of the tower of Babel by Pieter Bruegel, 1563
Oil on oak panel, Kunsthistorisches Museum, Vienna

http://www.towerofbabel.com