Towards an Ontology for Software

**Design:** The Intention/Locality Hypothesis

Amnon H Eden, Raymond Turner
Department of Computer Science, University of Essex, UK

*Europe—Computing and Philosophy (ECAP) 2005*
Västerås, Sweden, 2-4 June 2005

---

**Context**

- "Software"
  - Source-code
- "Software design"
  - Statements about programs ("meta"-statements)
- Ontological investigation
  - Examine, distinguish, classify, subject to scientific analysis (Quine)

---

**Summary: Contributions**

1. The top-level ontology for software design is
   - Strategic
   - Tactical
   - Implementation

<table>
<thead>
<tr>
<th>Design statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strategic</td>
</tr>
<tr>
<td>Tactical</td>
</tr>
<tr>
<td>Implementation</td>
</tr>
</tbody>
</table>

2. This ontology is formalized by the Intension/Locality criteria

<table>
<thead>
<tr>
<th>Design criterion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strategic</td>
</tr>
<tr>
<td>Tactical</td>
</tr>
<tr>
<td>Implementation</td>
</tr>
</tbody>
</table>

---

**Abstraction in Computer Science**

- Abstractions play central role in CS
- Means to—
  - Model complex worlds and systems
  - Aid problem solving
  - Manage complexity
  - Move away from implementation issues
Software: Two levels of abstraction

Design
(Statements about programs)

Programs

Abstraction in Programming Lang.

- Programming languages evolved by enriching the ontology with abstractions of ever-higher level
  - Matured & established over the years
  - Relatively well-defined ontology ("paradigm")
  - Formal semantics (e.g., denotational/operational)

<table>
<thead>
<tr>
<th>Language</th>
<th>Abstraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fortran</td>
<td>iterations, arrays</td>
</tr>
<tr>
<td>Algol, Modula</td>
<td>recursion</td>
</tr>
<tr>
<td>Smalltalk, Java</td>
<td>objects</td>
</tr>
</tbody>
</table>

Abstraction in software design

- The “Tower of Babel syndrome”: Ontological gluttony
  - No common language
  - No uniform formalism
  - No common ontological commitments
  - No overall conceptual perspective

Objective

- To “structure” the space of all design statements
- Refine “design” into 3 abstraction strata
  - Informal ontology: Strategic, tactical, implementation
  - Formalized ontology: The Intension/Locality hierarchy
What are design statements?

- Talk about programs (“meta” statements)
- Represent organizational and behavioral properties of programs
- Three examples:
  - **Architectural style:** Client-Server
  - **Data structure:** Stack
  - **Program documentation:** JNDI specification

Example: **Broker architecture**

- An **Architectural style**, defined by the following principles:
  - **Components:** “Servers” and “clients”
  - **Connector:** Broker + network

Example: **Stack**

- A **Data Structure**, defined by operations:
  - `push(x,s)`: Adds object x to “top” of s
  - `pop(s)`: Removes last object “pushed” into s
  - `top(s)`: Returns last object “pushed” into s
  - `empty(s)`: Returns true if s empty, false otherwise

```
public class Stack {
  private Object top;
  private Object[] data;

  public Stack(int capacity) {
    data = new Object[capacity];
  }

  public void push(Object x) {
    data[++top] = x;
  }

  public Object pop() {
    if (top < 0) throw new EmptyStackException();
    return data[top--];
  }

  public boolean empty() {
    return top < 0;
  }
}
```

Example: **Program documentation**

- Describing specific details of a particular implementation:
  - “JNDI defines the Attribute interface ... An attribute consists of an attribute identifier (a string) and a set of attribute values...”

```
public class Attribute {
  public DirContext getAttributeDefinition() {
    throws NamingException;
  }

  public DirContext getAttributeSyntaxDefinition() {
    throws NamingException;
    ...
}
```
Summary: 3 levels of abstraction

- Broker
- Stack
- JNDI

Questions about design statements

- Is this space discrete or continuous?
- Are these differences qualitative or quantitative?

Discrete (qualitative) vs Continuous (quantitative)

Answer: Discrete (qualitative)

- There are qualitative differences between these statements.
- This is the top-level ontology for software design!

Can we formalize this ontology?

- Strategic statements
  - Expressing global design properties
  - Decided early in the software lifecycle
- Tactical statements
  - Expressing local design properties
  - Decided late in the software lifecycle
- Implementation statements
  - Expressing details of the implementation
  - Part of the implementation (the latest stage in the software lifecycle)
The Intension/Locality criteria

- The Locality criterion: A statement is local if and only if it is preserved under expansion.
- The Intension criterion: A statement is extensional if and only if it is preserved both under expansion and under reduction.

(Terminology: Model-theoretic)

The I/L Hypothesis

- The Intension/Locality hierarchy coincides with the top-level ontology for statements in software design:
  - Strategic ∈ M
  - Tactical ∈ L
  - Implementation ∈ E

What does it mean?

- Strategic statements ∈ M:
  - Architectural styles
  - Programming paradigms
  - Component-based software engineering standards
  - Application frameworks

- Tactical statements ∈ L:
  - Data structures
  - Design principles
  - Design patterns

- Implementation statements ∈ E:
  - UML: class/collaboration/package diagrams
  - Software documentation