

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

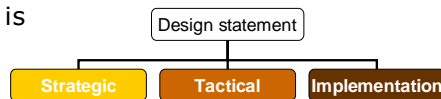
Context

- “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**



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



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

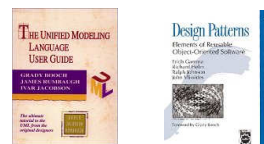
Abstraction in Programming Lang.

- 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)

Fortran	<i>iterations, arrays</i>
Algol, Modula	<i>recursion</i>
Smalltalk, Java	<i>objects</i>

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

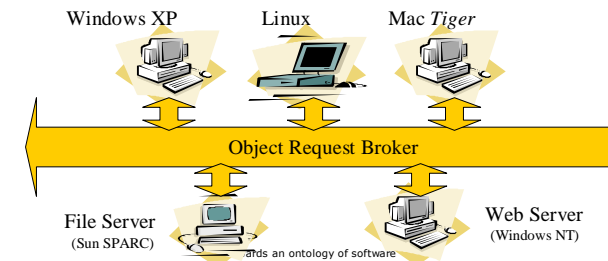
- To "structure" the space of all design statements
- Refine "design" into 3 abstraction strata
 - ◆ Informal ontology:
Strategic, tactical, implementation
 - ◆ Formalized ontology:
The Intension/Locality hierarchy

What are design statements?

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

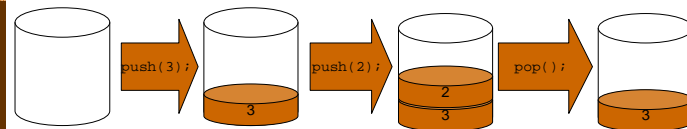
Example: *Broker architecture*

- 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

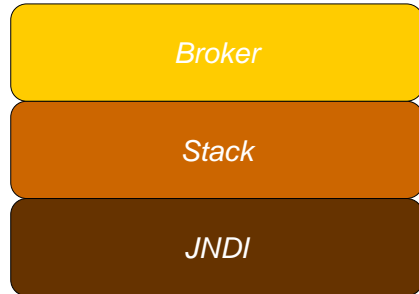


Example: *Program documentation*

- 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



Towards an ontology of software

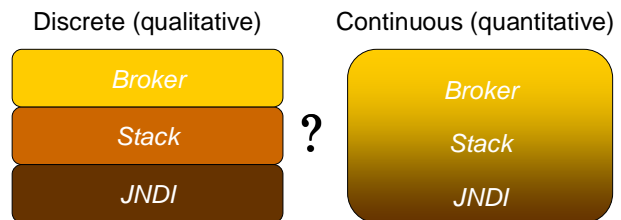
13

Questions about design statements

- *Is this space discrete or continuous?*

Also,

- *Are these differences qualitative or quantitative?*



Towards an ontology of software

14

Answer: Discrete (qualitative)

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

Strategic statements

- Expressing *global* design properties
- Decided early in the software lifecycle

Tactical statements

- Expressing *local* design properties
- Decided late in the software lifecycle

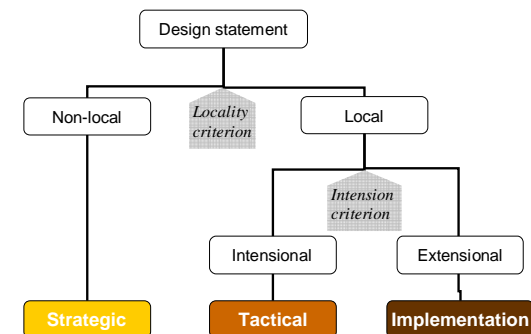
Implementation statements

- Expressing *details* of the implementation
- Part of the implementation (the latest stage in the software lifecycle)

Towards an ontology of software

15

Can we formalize this ontology?



Towards an ontology of software

16

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

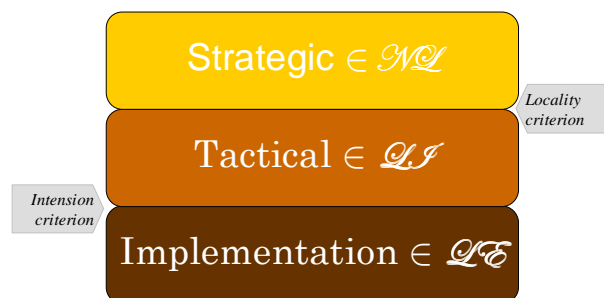
(Terminology: Model-theoretic)

The Intension/Locality hierarchy



The I/L Hypothesis

- The **Intension/Locality hierarchy** coincides with the top-level ontology for statements in software design:



What does it mean?

- Strategic statements \in NL
 - ◆ Architectural styles
 - ◆ Programming paradigms
 - ◆ Component-based software engineering standards
 - ◆ Application frameworks
- Tactical statements \in LI
 - ◆ Data structures
 - ◆ Design principles
 - ◆ Design patterns
- Implementation statements \in LE
 - ◆ UML: *class/collaboration/package* diagrams
 - ◆ Software documentation