A Structuralist View on Object-Oriented Thought Paradigms by Employment of CLOS

Lennard Wolf
lennard.wolf@student.hpi.de

Hasso Plattner Institute
Prof.-Dr.-Helmert-Straße 2-3
14482 Potsdam
Germany

Abstract. A Structuralist View on Object-Oriented Thought Paradigms by Employment of CLOS

1 Introduction

The following is a summary of each section. In section...

2 Context

In this Section we will give an overview of the history of CLOS, then we will introduce the ideas behind Lisp and Object-Orientation in general, and finally we will introduce the three main concepts in CLOS that make it unique.

2.1 Object-Orientation

- OO = objects + classes + inheritance
- Objects have attributes (state) and behaviour
- These Objects can interact with one another by sending messages
- Classes are code templates with initial values for state and implementations of behavior
- Inheritance: deriving more specific classes from others
- Most popular class-based: Objects are Instances of Classes
- Gives programmers the ability to model the real world

2.2 Lisp

- List Processor: Everything is a list (plus 3 4)
- No Difference between data and code
- Use Lisp Macros to easily extend Lisp
- Therefore it is easy to create Domain Specific Languages etc.
- possible to add OO to Lisp in Lisp
- explain Macros in detail

2.3 History of CLOS

- Many implementations ("Tower of Babel" situation) (Flavors, CommonLoops)
- 1986: Workgroup of different researchers from Xerox PARC (CommonLoops) and Symbolic Inc. (New Flavors) worked on it together
- CLOS: System built on top of Common Lisp
- meta-object protocol consisting of 33 functions & 8 macros
- CLOS is portable to different LISP implementations

2.4 Main Concepts

Generic Functions blablablablablablablablablablablabla

Multiple Inheritance blablablablablablablablablablablabla

Method Combination blablablablablablablablablablablabla

Fig. 1. test

3 Problem

- how can we build a people description interface that is easily extendable etc., like DSL (?) for people and their positions
- using langs like Java (?) might be problematic, give example

4 Approach

- 3 concepts will be used
- maybe give function that lets user add classes

5 Implementation

- give code details for concepts from Approach

6 Evaluation

- DSL type declaration is easily maintainable
- using langs like Java (?) might be problematic, give example

7 Conclusion

Write me pl0x