The Culture of Programming Languages

Sebastian Fleissner Elisa Baniassad

Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
{seb, elisa}@cse.cuhk.edu.hk

Abstract

The theme of this workshop is the proliferation of ideas about calling into question the cultural roots of our current programming languages, and the search for alternative paradigms with other cultural bases.

Categories and Subject Descriptors D.1.0 [*Software / Programming Techniques*]: General

General Terms Languages

Keywords Culture, Programming, Software Engineering

1. Introduction

Software design and development techniques, such as object-oriented programming, are strongly influenced by the reasoning and thought patterns prevalent in the society they were created in. Cognitive science and evolutionary psychology assume that human cognition is universal or, in other words, that individuals from different cultures all share the same basic reasoning styles and thought patterns.

However, cultural psychologists like Nisbett posit that people from different cultural backgrounds think differently. In his book "The Geography of Thought" Nisbett shows that "Eastern" (predominantly Chinese and Japanese) and "Western" reasoning styles differ greatly: Westerners tend to focus on objects, whereas Easterners focus on fields of interaction.

The goal of this workshop is to create awareness of the relationship between programming paradigms and cultural reasoning and thought patterns, to share ideas, and to discuss possible research topics.

2. The Origin of Object-Orientation

Aristotelian physics [1], a theory developed by the ancient Greek philosopher Aristotle in the year 350 BCE , suggests that gravity can be considered to be an attribute that resides within objects. Heavy objects, such as stones and rocks, possess a stronger gravity attribute than lighter objects. In fact, very light objects like gas and air have a "levity" attribute instead of a gravity attribute. Since gravity is considered as an attribute of objects and not an outside force, Aristotelian physics posits the view that the world can be understood and described as a collection of more or less independent objects that can be described and analyzed through categorization and formal logic.

The approach of focusing mainly on objects and their attributes while ignoring possible outside forces is very common and encouraged in the context of traditional object-oriented software design and development: Objects are isolated from their environment as much as possible and static relationships between objects are defined in advance.

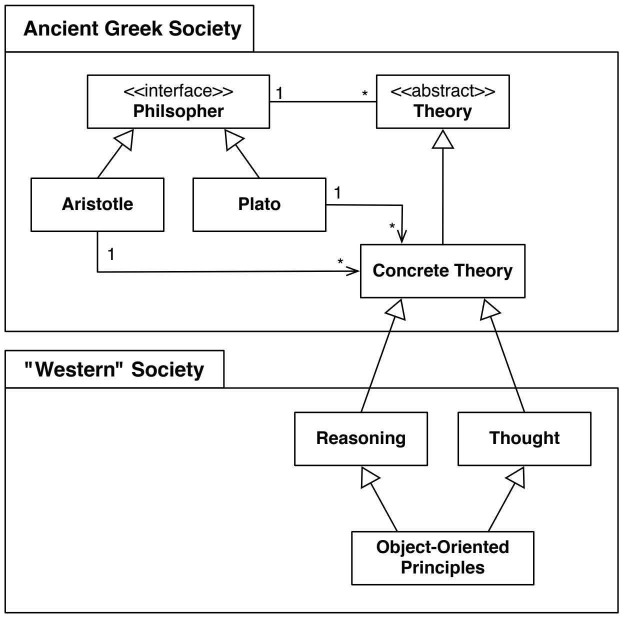

Figure 1. The Origin of Object-Orientation (in UML)

The theories by Aristotle and other ancient Greek philosophers like Plato are both a mirror of and influence on the style of reasoning and thought prevalent in the so-called

OOPSLA 2009, October 25–29, 2009, Orlando, Florida, USA.
ACM 978-1-60558-768-4/09/10.

Aristotle's Theories	**Object-Orientation**
Consider the world as a static and unchanging collection of objects.	Describes programs using a set of predefined objects with static relations.
Analyze objects and their attributes in isolation from their context.	Isolates objects as much as possible from their context.
Relevance of possible outside forces is ignored.	Outside forces are mitigated.

Table 1. Aristotle, Plato, and Object-Orientation

"Western" societies, such as Europe and North America, where object-oriented programming and most other popular programming paradigms originated. Figure 1 uses the object-oriented approach to illustrate the relationship of theories originally formulated by philosophers like Aristotle and Plato, their influence on "Western" reasoning and thought, and the principles of object-orientation. Table 1 summarizes similarities between Aristotle's theories and object-orientation.

3. Initial Work

Ancient Chinese philosophers did not focus on objects and their attributes, but rather considered the broad context and saw the world in terms of harmony, context, roles, obligations, and resonance. For example, a person was considered not as an individual with a constant unique identity, but rather as a member of several collectives. As described in [4], ancient Chinese philosophers and people saw the world as a mass of continuously interacting substances rather than a collection of discrete objects. Each substance and every event in the world was considered to be related to every other event.

At OOPSLA 2006, we presented a paper called the "Geography of Programming"' [2] that outlined our initial vision for alternative programming languages based on the ideas found in Chinese philosophy, which lend themselves better to description of interactions between entities. In this paper we present a study, in which we interview Chinese participants about how they would describe a typical object-oriented scene, and then attempt to capture and distill their descriptions into the guidelines for an alternative programming paradigm.

In our OOPSLA 2008 paper "Towards Harmony-Oriented Programming" we introduced our work towards proposing a concrete software development approach based on principles found in Chinese philosophy.

The main idea behind harmony-oriented programming (HOP) is that pieces of a program always interact with their environment as a whole and usually not with other program parts directly. Table 2 illustrates important conceptual differences between harmony-oriented software programming and object-oriented programming (OOP). Harmony-oriented programming challenges established and widely accepted object-oriented design principles, such as strong encapsulation, information hiding, and inheritance, and favors more flexible and ah-hoc approaches for structuring and implementing programs.

Apart from proposing harmony-oriented principles, our recent work focuses on constructs of harmony-oriented programs and a Smalltalk-based runtime and development environment.

Object-Orientation	**Harmony-Orientation**
Individualism	Holism
Explicit Boundaries	Fuzzy Boundaries
Explicit Relationships	Implicit Relationships
Protocols / Negotation	Observation

Table 2. Object-Orientation and Harmony-Orientation

4. About the Organizers

Sebastian Fleissner recently received his Ph.D. from the Chinese University of Hong Kong where he was working with Elisa Baniassad. He has published several papers at Onward! in the area of harmony-orientation and self-sustaining systems.

Elisa Baniassad is an assistant professor at the Chinese University of Hong Kong, and has been actively involved in Onward! for several years. She is very interested in promoting the consideration of fringe programming models.

References

[1] Aristotle. *Physics*. 350 BCE.

[2] E. Baniassad and S. Fleissner. The geography of programming. In *OOPSLA 2006: Companion to the 21st annual ACM SIGPLAN conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 560–573. ACM Press, 2006.

[3] S. Fleissner and E. Baniassad. Towards harmony-oriented programming. In *OOPSLA '08: Companion to the 23rd ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 819–822. ACM Press, 2008.

[4] C. Hansen. *Language and Logic in Ancient China*. University of Michigan Press, 1983.