HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Studiengang
Informatik

Öffentliche Vorträge

Dienstag, 17. Juni 2014

Talk 1, 17.00 – 17:45 Uhr, Gebäude 3, Raum 3.008
Talk 2, 17:50 – 19:00 Uhr, Gebäude 3, Raum 3.008

Anschliessend für **SWEN Mitglieder** gemeinsames Barbecue am See mit dem C++ Komitee (Anmeldung an ifs@hsr.ch)

SWEN Talks

Talk 1: Standardization process and C++14

JC van Winkel (Google)

Talk 2: Value Semantics: It aint't about the syntax

John Lakos (Bloomberg)

Talk 1, 17.00 – 17:45 Uhr

Standardization process and C++14 (JC van Winkel)
In this talk, JC van Winkel will look at the standardization of C++ and the future of C++. How does the process work? What can we expect? We will also look at new features that are specified in the Draft C++14 standard, and what lies ahead in "Technical Specifications" that are in the pipeline waiting.

Referent JC van Winkel
He has an M.S. in Computer Science. From 1990 to 2010 he worked at AT Computing, a small courseware and consulting firm in the Netherlands. There he taught UNIX and C++. Since 2010 he is working in Google's Site Reliability Engineering department in Zurich. JC is head of the Netherlands delegation in the ISO C++ standards committee since 1997.

Talk 2, 17:50 – 19:00 Uhr

Value Semantics: It aint't about the syntax (John Lakos)

When people talk about a type as having *value semantics,* they are often thinking about its ability to be passed or returned by value. In order to do that, C++ requires the type to have a copy constructor, and so people routinely implement copy constructors on their classes, which begs the question,

"Should an object of that type be copyable at all?" If so, what should be true about the copy? Should it have the same state and behavior as the original object? What does copying an object mean?!

SWEN
Software Engineering Network

IFS INSTITUTE FOR SOFTWARE

By *value type*, most people assume that the type is specifically intended to represent a member of some set (of values). A value-semantic type, however, is one that strives to approximate an abstract *mathematical* type (e.g., integer, complex-number), which comprises operations as well as values. When we copy an object of a value-semantic type, the new object might not have the same state, or even the same behavior as the original object; for proper value semantic types, however, the copy will have the same value.

In this talk, we begin by gaining an intuitive feel for what we mean by *value* by identifying *salient attributes*, i.e., those that contribute to value, and by contrasting types whose objects naturally represent values with those that don't. After quickly reviewing the syntactic properties common to typical value types, we dive into the much deeper issues that value semantics entail. In particular, we explore the subtle Essential Property of Value, which applies to every *salient* mutating operation on a value-semantic object, and then profitably apply this property to realize a correct design for each of a variety of increasingly interesting (value-semantic) classes.

Referent John Lakos
He is the author of "Large Scale C++ Software Design.", serves at Bloomberg LP in New York City as a senior architect and mentor for C++ software development world-wide. He is a member of the C++ Standards Committee. Previously, Dr. Lakos worked at Bear Stearns and Mentor Graphics. He holds Ph.D. in Computer Science ('97) and an Sc.D. in Electrical Engineering ('89) from Columbia University. Dr. Lakos received his undergraduate degrees from MIT in Mathematics ('82) and Computer Science ('81). His next book, entitled "Large-Scale C++, Volume I: Process and Architecture", is anticipated in 2014.