3 Abstracts of Presentations

The following abstracts appear in alphabetical order of the speakers.

Multi-Aspect System Descriptions for Object-Oriented Programming

Keijiro Araki[1]
Kyushu University
Fukuoka, JAPAN

We report our case study of a system development with a variety of descriptions and analyses using Z, ML and Smalltalk. We get a set of system descriptions from various viewpoints including abstract functional aspects, system structural aspects, implementation feasibility aspects, and so on. We need not necessarily start from an abstract formal specification and refine it to a final concrete program, but we may start wherever easy to start. We discuss the roles of such descriptions and their interrelationships. Especially, we use ML as an executable specification language. By describing a system in ML, we would get insigts for abstract formal specifications as well as for system architectures and design issues. We intend to accumulate much experience in system development with a set of various descriptions and build up a road map for system development based on formal methods.

ETOILE-Specifications and Real-Time issues

Gilles Bernot[2]
Université d'Evry
Evry France

ETOILE is an object based formal specification theory. After a short introduction about the way ETOILE-specifications are structured, we describe the syntax of the specification of object types and we show how they can be combined to specify object systems.

An object type specification can be described as a sort of "star", the center of which is the type of interest, the branches being types of objects that can be used by an object of the center. ("etoile" is the French translation of "star"). A system of objects is then obtained by putting together several such stars. We follow the principle to match each branch of a star with the center of another one.

Then we show that adding a new object to an already existing system can entirely modify the system properties. Thus, if we want to establish formally the properties of

[1]This work has been done with Han-Myung Chang and Toshiyuki Tanaka.

[2]This is joint work with Marc Aiguier and Stefan Beroff.

an object system, we cannot proceed by establishing some lemmas on a small system, and complete the lemmas incrementally after adding objects one by one to the system, until we reach the full system under interest. Consequently, an incremental proving method cannot be obtained this way.

To allow to establish properties on an incremental way, we propose instead a method based on "object refinements". Within our ETOILE theory, we have defined a theory of refinement which has the interesting property that: if a system SYS1 correctly implements an object type O, and if SYS2 is a system that contains O and satisfies a property phi, then the system SYS3 obtained by replacing O by SYS1 in SYS2 still satisfies phi. This allow to start from a small, very abstract system with a small number of very abstract object types; and to make the system incrementally bigger and more precise, by successive refinements of its object types.

Real-time aspects are also currently an important topic for ETOILE. ETOILE specifications are used since 5 years to specify hardware/software systems for the co-design of some telecommunication systems and we often need to introduce statements about some delays in actual nanoseconds.

We have only defined such real time formulas for systems made of a unique object (i.e. systems with a unique global state). The idea is to add a special data type which represents time durations, with some built-in operations and predicates. This allows to specify methods that modify dynamically their behaviours according to their own execution time for example. An example is fully described to illustrate the approach.

Object orientation in domain analysis for reuse

Alfs Berztiss
University of Pittsburgh
Pittsburgh, USA and
University of Stockholm
Stockholm, Sweden

We consider domain analysis in the context of reuse-based development of software systems. Domain models are expressed in terms of processes, and a process is defined as an ordered set of tasks. A generic task is first formulated as a cliche in natural language, and reuse is achieved by adapting it for several specific applications. We follow an object-oriented approach to the definition of processes, with the understanding that object orientation has two aspects. One relates to data, but there is also a process aspect, and this latter aspect is our primary concern here. We consider process models and object orientation in reuse-driven development of information systems and control systems. Our method of defining application domains in terms of generic processes, where a process is regarded as a collection of tasks, is used to define the domain of repairs. This domain includes repair of machinery, road repirs, surgical procedures, and software debugging. We specialize the generic process for the last application. We also consider situations, by which we mean tasks that are so general that they arise in several domains.