To appear in ESUG Newsletter, December 1993
(ESUG: European Smalltalk Users Group)

Some Impressions on OOPSLA'93

Jean-Pierre Briot
LITP, University Paris VI - CNRS, Paris , France
briot@litp.ibp.fr
& Dept. of Information Science, the University of Tokyo, Japan
briot@is.s.u-tokyo.ac.jp

OOPSLA'93 has been held in Washington DC, USA, from 26th September to 1st October 1993. In this text, we attempt at summarizing what we found interesting there. This summary is by no means exhaustive, one body being not enough to attend all events!

We will start with a few statistics showing that OOPSLA is still expanding its variety of events. The technical program included 26 papers. There were also 3 invited talks, one special track on experience reports, plus one on education, and 10 panels(!). There were 35 tutorials, 19 workshops, 26 demonstrations and 78 exhibitors. Also present were posters and birds of a feather sessions (among them, a short ESUG meeting organized by Mario Wolczko). The technical program was of very good quality (the statistics show that 26 papers were accepted from 278 submissions). Meanwhile, the proceedings being widely available along ACM Sigplan Notices, we will rather focus on demonstrations and exhibits showing new Smalltalk softwares. Note that without time and opportunity for testing following softwares, these are impressions, not real evaluations.

The general setting of this survey is that Smalltalk is doing very well! One clue is that a new Smalltalk system is born early this year. Its name is SmalltalkAgents™, from the Quasar Knowledge Systems company (in Bethesda MD, USA). This is a brand new Smalltalk running on Macintosh. SmalltalkAgents is supposed to be deeply integrated within the Macintosh features and offers numerous Macintosh toolbox routines. (Actually Quasar states there will be eventually a RISC version, but one may wonder about the exact compability between the two versions). There are abilities to refer to Macintosh data structures, to convert them to Smalltalk data and vice-versa, and finally to call and be called from foreign methods. The scheduler is preemptive and all collection classes are supposedly protected against concurrent accesses. About collections, note that there is a class List with built-in list operators ({ and }) to dynamically create lists. Libraries offer unique name spaces to avoid name clashes. The look and feel of the programming environment tools is quite different from Smalltalk-80 and V. There are interesting shortcuts like the bookmarks, in order to avoid opening numerous browsers while you explore libraries. The user interface is event-driven (and not based on polling controllers). After reviewing some of the main features, it is too early to draw definite conclusions with too little demonstration during OOPSLA. But this looks like solid stuff, and not just a plain copy of other systems. The class libraries are not exactly similar to other systems, and some are very differently organized (e.g., user-interface libraries). There are many shortcuts, e.g., to access indexed variables of objects or to call external methods in a goal of efficiency and access to low-level resources. This is obviously a product intended for real developement, not just for pedagogical purposes! The system including libraries and sources was listed around $500 U.S.

Other interesting Smalltalk softwares are following. Strongtalk™ is a type checking pre-processor for Smalltalk. It is a company product (Horizon Technologies of New York). A paper in the main proceedings describes it and the system was also demonstrated. Strongtalk looks convincing as a good tradeoff between completeness and practicality. In the Strongtalk system, type information is used for type checking, not for compiler optimization (although this could obviously be used in future by some enhanced compiler). Type analysis is done locally, so that one may still incrementally develop programs. The type declarations follow standard conventions. Strongtalk supports parameterized and polymorphic types, as well as dynamic type checking. Special type identifiers like Self, respectively Instance, are interpreted as the type of the receiver, respectively of its instances. Strongtalk help retaining the flexibility of Smalltalk. Moreover the typechecker is integrated within the standard environment, allowing incremental declarations and type checks while still using standard (untyped) libraries.

Object Explorer™ is an oncoming product, designed by Kent Beck (now consultant, First Class Software) which was also demonstrated. This may be described as a graphical inspector focusing on relationships between objects. It shows the components of an object and the result of messages sent. It clearly visualizes if there are references to a same object. One may hide some object internals to focus on others. Explorer automatically computes and visualizes contracts, that is the set of messages that a component object should implement. This is particularly useful while building or replacing components. It looks as a simple but powerful tool to explore and understand pieces of programs and to help interactively building new ones. For instance one does not need to worry about having to write down all initialization methods at first, but one may initialize objects interactively and quickly through Explorer while in the development process.

Last interesting demonstration is about a university project software (Duane Szaffron, U. of Alberta, Edmonton AL, Canada), named Enterprise, which helps distributing coarse grain parallel computations among a network of workstations. Enterprise runs in Smalltalk and provides a graphical view of various components of computations. Components need to be identified through decomposition of the program in several independent modules (e.g., animation programs with a pipeline of various stages for rendering until producing the frames). They are written in standard C code with minimal communication interface with other components. (Communication is asynchronous or eager reply/wait by necessity in order to exploit parallelism.) Enterprise allows to arrange and rearrange aggregation of components (e.g., along a division or a pipeline flow) and possibly duplicate components in case of bottlenecks. Animation is used to monitor and help debugging such programs. The demonstration was very convincing and although the programmer still needs to decompose its application in independent modules, the system makes the allocation and setting transparent and helps profiling the computation.

There was also a demonstration of Self, the prototype-based Smalltalk-alike system that Dave Ungar and his team develop at Sun Labs. I should confess that the demonstration was very disappointing to me. Although there has always been great work on optimization techniques, the language and environment aspects are clearly not a match. The ability to add new slots onto any object was described as a plus over Smalltalk. I rather find the opposite in that it may easily lead to very messy programs and semantics. As an illustration the screen became quickly full/messy of little rectangles connected by various lines although the example was still very trivial (planar points). It looks like there has not been any real development of Self programs besides the system itself within the team. Meanwhile there were some interesting ideas borrowed from animation practices in order to improve movements of graphical objects, e.g., windows being closed behave like cartoon characters by visualizing starting and trajectory!

As other main exhibitors, Digitalk and OTI were present. As a paradox, the main Smalltalk vendor, ParcPlace, was not there (Adele Goldberg was there as an individual). I was told (by other people) that they are going on stock market soon, so they wanted to minimize spendings in order to show best financial records. Also note that among the three IBM booths, two were showing Smalltalk-based products, one being OTI Envy/400, the Envy™ development environment for IBM AS/400 computers, and Visual Age, an interface builder.

In conclusion, this little review of some of the Smalltalk activities at OOPSLA'93 showed the liveness of the Smalltalk community and industrial activity. Another clue was the number of jobs offered for Smalltalk programmers! The global impression from the conference is that there are now two main object-oriented languages: C++ and Smalltalk. Some others have found their small niche (Eiffel, Beta). Some others have drastically declined industrially speaking (CommonLisp/CLOS). It looks like, as one may expect, the scientific C-based software communities is still moving to C++. But commercial Cobol-based communities (e.g., finance business) are moving directly to Smalltalk. More and more people are getting the experience or the knowledge that Smalltalk is a simpler language and a wonderful development tool, and that C++ is too large and complex language. (I was personally astonished when reading the titles and abstracts of tutorials to be presented at C++ World, the alleged "largest C++ event in '93". Almost a third of the tutorials talk more or less explicitly about how to avoid traps! The issue is that most of software or methodological solutions proposed look incomplete or/and complex, thus adding up to the difficulty of using properly the language!). So we may be optimistic for the future of Smalltalk!

Thanks to Jeff and Mario for sharing comments and tips about the conference.