Re-inventing old mistakes?

Benedict Heal

Way back in the 60's we grew concerned that many applications contained different ways of representing and processing what was essentially the same information. Up went the cry 'separate data from processing' and databases were invented to hold a single represention of the information. We began to talk about 'running multiple programs against the same data', and even stopped using the word 'information' as we became aware that a database column name didn't really define the intended semantics of a data field. The column name 'month' , 'balance' or 'telephone' told us very little about permissible data values, or allowed operations. We came to rely on general intuitions to define what was and what was not a reasonable data manipulation.

After some 20 years practice in trying to design data independent of its processing, we began to abandon the attempt, invented objects, and put back with the data just enough processing to give the data meaning, while application dependent processing was placed in application dependent objects. It was hard work, but after 20 years we began to get quite good at separating concerns, and could tell how much of the behaviour really was part of the semantics of the object being manipulated, and how much was part of a particular application. There were still lots of problems, such as how one made objects persistent, or how one could transmit objects over networks.

We then had the brilliant idea of separating object state from processing, and invented XML. We started abandoning hard-learned lessons about encapsulation, and merrily shipped direct representations of internal object state around the network. We gave up any attempt at defining what data fields meant. We went back to thinking of dates as triples of integers, where the permitted values and interpretations of each integer could be guessed from a tag name and some intuitions. We even tried to hoodwink people into overlooking what we had done, by using fancy words like 'semantics' to mean merely that we had actually specified a simple syntactic grammar of our data, and we started using the vocabulary of 'data exchange' where once we talked about transmitting objects.

Sometimes this is defended by saying that an application program can easily provide a simple mapping from a tag name to a class name, and that the class will provide all the semantics. This could be true if the mapping and class were part of the XML, but if they are not, any receiver of the data can provide any interpretation.

Some of the things that I want to learn before I trust myself to make new mistakes with XML, rather than just repeat the old ones, include:

- should XML represent concrete object states, or abstract object states?

- how should I attach to XML a minimal required semantics that will suitably restrict the interpretations that the receiver can put on the data?

- have I mis-understood the whole game, and remaining stuck in classical object-speak, blinded myself to an exciting new computing paradigm?