Editors' Corner

John Pugh

Paul White

HERE'S A TECHNICAL QUESTION FOR YOU—do you ever use **become:** in your applications? What twigged the issue was the fact that, for a long time now, our training materials have included an application that makes use of the **become:** operation. It was written to illustrate the power of Smalltalk and to introduce people to the issue of *object mutation*. But we know that in our own software development, we rarely use it; we're sure that's true of others as well. The question is, why not? And if it's not to be used, then should vendors continue to support it?

For those of you who may not be familiar with **become:**, it is defined as an instance method in the class Object. The intention of the message send "object1 become: object2" is to mutate object1 into object2. However, there are two different implementations found in the various Smalltalks. Visual Smalltalk's implementation has always had the effect that all references to object1 are made references to object2; object1 is left with no references and is therefore garbage collected. IBM Smalltalk for Windows and OS/2 provide similar behavior. VisualWorks, on the other hand, implements a **become:** that swaps the references between the two objects—all the original references to object1 become references to object2, and all the original references to object2 are made references to object1; neither object gets garbage collected.

Here's a technical question for you—do you ever use **become:** *in your applications?*

This, of course, leaves you with radically different behavior for the same message, which is one obvious reason for not using **become:**. The classic example of this problem is someObject become: nil yields radically different results! Another reason for the lack of use of **become:** is historical. Early implementations of Smalltalk did not always yield the expected results and/or perform well at all. As a result, many gave up on the operation and found another approach.

But the question still lingers—even if the issues listed were resolved, would it be appropriate to make use of **become:**? The answer, of course, is "it depends…". The real issue is there are two different interpretations of what **become:** really means in an application. A statement such as "with **become:**, you can make an elephant become a mouse" is not appropriate because the two objects are not related to each other in any manner. This really represents replacement, not mutation.

Another interpretation of **become:** is that it provides a mechanism to allow an object to dynamically change its behavior. Consider this statement: "As a boy grows older, he becomes a man." The message **become:** can be used in this context to model the changing behavior of a person. For example, mobility is achieved by a young child through crawling; an older child walks. As a result, when the message **move Around** is sent to a person, the message is interpreted differently at different ages. A possible implementation would be to have a child object actually mutate into an adult object. It is important that the object decides to do this itself and does not depend on other objects. This avoids code like self **age** < 2 **ifTrue**: [self **crawl**] **ifFalse**: [self **walk**]. Although the example is trivial, it is illustrative of the type of behavior changes we require from an object.

So, is **become:** necessary? Probably. Applications do exist where it would be beneficial and perhaps even appropriate. But in the end we find it is a neat idea, but for most problems we should find another solution. For example, in the person example cited, it's always possible to introduce a generic Person object and have them have roles or characteristics. So, a person may either have child or adult characteristics and its behavior is directly dependent on which characteristics it has. These are nontrivial systems to implement, but they give you the desired behavior.

As this is the Smalltalk Solutions issue, we hope many of you take advantage of the conference. It's the best opportunity for Smalltalk to show off its successes. Please drop by and introduce yourself!