Subject: Re: why not "T & Add(const T & x)" in all containers
Posted by rylek on Wed, 08 Sep 2010 08:24:30 GMT
View Forum Message <> Reply to Message

Hi there!

It is a pleasure to play the third party, though the semantics of the term are highly disputable here. I think it's possible Mirek originally didn't include the return reference in Vector::Add to emphasize the fact that such references into Vectors are volatile in principle (specifically they are periodically invalidated by the Add function itself while reallocating the physical Vector data). But, of course, the same argument holds for Add() which does return the reference. Moreover, Array::Add(T *newobj) also returns the reference, albeit for different reasons.

U++ also decidedly avoids returning references in pick assignment operators, which is natural because a "chain" assignment (a = b = c) in such cases exhibits undesirable behaviour (by destroying b). But Vector::Add(const T&) doesn't have this problem; the only thing that has to be avoided is rather artificial constructs of the form

```
vector.Add(vector.Add(obj))
```

exactly because of the periodical Vector reference invalidation. But then again you can run into exactly the same problems by writing, e.g.

```
vector.Add(vector[5]);
```

so that this is no specific of Vector::Add(const T&) either. To sum it all up, I currently see no practical reasons against modifying Vector::Add to include the return reference. I would rather say that it's like updating old code to match interface standards adopted / developed later on, in fact I believe Vector is one of the very oldest things in U++ (although it's been rewritten quite a few times since its inception).

Regards

Tomas