Doc No: X3J16/97-0065 WG21/N1103
Date: July 16, 1997
Project: Programming Language C++
Ref Doc:
Reply to: Josee Lajoie
(josee@vnet.ibm.com)

Core 1 - Object Elision

Replace 12.8, para 15 with:

Whenever a temporary class object is copied using a copy constructor,
and this object and the copy have the same cv-unqualified type,
an implementation is permitted to treat the original and the copy as two
different ways of referring to the same object and not perform a copy at all,
even if the class copy constructor or destructor have side effects.
For a function with a class return type,
if the expression in a return statement for this function is the name of a
local object,
and the cv-unqualified type of the local object is the same as the function
return type,
an implementation is permitted to omit creating the temporary object to hold
the function return value,
even if the class copy constructor or destructor has side effects.
In these cases, the object is destroyed at the later of times when the original
and the copy would have been destroyed without the optimization.*f
.Fs
Because only one object is destroyed instead of two, and one copy constructor
is not executed, there is still one object destroyed for each one constructed.
.Fe
.E[
.Cb

```
        class Thing {
        public:
                Thing();
                ~Thing();
                Thing(const Thing&);
                Thing operator=(const Thing&);
                void fun();
        };
```

.Ce
.Cb

```
        Thing f() {
                Thing t;
                return t;
        }

        Thing t2 = f();
```

.Ce
Here
.CW t
does not need to be copied when returning from
.CW f .
The return value of
.CW f
may be constructed directly into the object
.CW t2 .
.E] e