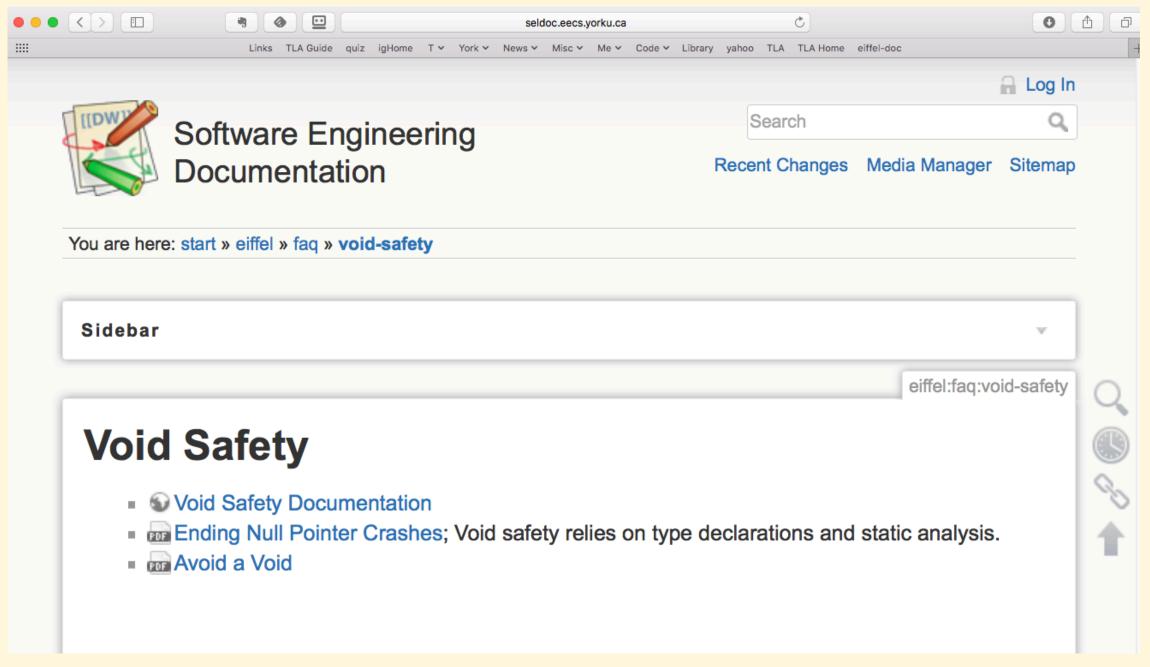
## **Void Safety**

"Null references: The billion dollar mistake" (2009)

Turing Award Winner: Sir Tony Hoare



```
NODE[G] is the same as
                                       or a doubly-linked list stores:
NODE[G -> detachable separate ANY]
                                       reference to an element of a sequence; might be Void
                                       reference to the next node; might be Void
                                     a reference to the previous node; might be Voids
                      class
                                                            Public Queries
                          NODE[G -> detachable ANY]
                      creation
                                                            Export to {ANY}
                          make
                      feature
                          element : detachable G
                          previous: detachable NODE[G]
                          next:
                                     detachable NODE[G]
                      feature {NONE} -- Constructor
                          make(e: detachable G; p: detachable NODE[G]; n: detachable NODE[G])
                                  -- make a node with previous node `p' and next node `n'
                              do
                                  element := e
                                  previous := p
                                  next := n
                              ensure
                                   items_set: element = e and previous = p and next = n
                              end
```

end

## An ESpec unit test

```
Error List (310)
3 Errors 1 0 Warnings
Description
   VEVI: Variable is not properly set. Attribute(s): some_node
      Error code: VEVI
      Error: variable is not properly set.
     What to do: ensure the variable is properly set by the corresponding
       setter instruction.
     Class: TESTS
      Feature: make
     Attribute(s): some_node
     Line: 20
         do
      -> add_boolean_case (agent t1)
         end
```

```
some_node: NODE [STRING]

t0: BOOLEAN
    local
    do
        create some_node.make ("Yay!", Void, Void)
    end
```

## t1: BOOLEAN local

do

node: NODE[detachable STRING]

commute("t1: First test node")
-- c
te node.make (Void, Void, Void)
Resul
node.element ~ Void
node.previous = Void

node.next = Void

D. . . . darita .

Error List (110)

Description

Output

Description

Description

Description

Description

1 Error 0 Warnings

Error code: VUTA(2)

Error: target of the Object\_call might be void. What to do: ensure target of the call is attached.

Class: TESTS Feature: t1

Type: detachable NODE [detachable STRING\_8]

Line: 35

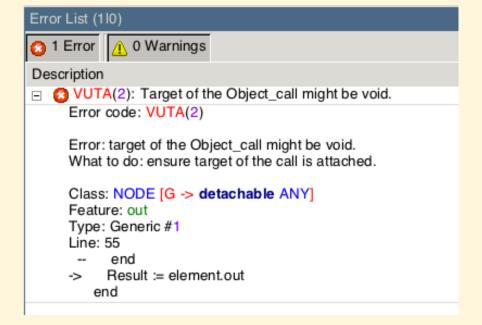
create node.make (Void, Void, Void)

-> Result := node.element ~ Void and node.previous = Void node: attached NODE[detachable STRING] attached by default

Must create node so that node.element is defined

```
Show meaningful text
class NODE[G -> detachable ANY] inherit
                                                                 in debugger
         ANY redefine out end
         DEBUG OUTPUT redefine out end
creation make feature
         element : G
         . . .
feature -- out
         debug_output: STRING
                            -- string representation for debugging
                  do
                            Result := out
                                                                Error List (110)
                  end
                                                                1 Error 0 Warnings
         out: STRING
                                                                 Description
                  do
                            if attached element as l_e ther 🗆 🗯 VUTA(2): Target of the Object_call might be void.
                                                                     Error code: VUTA(2)
                                     Result := l_e.out
                            else
                                                                     Error: target of the Object_call might be void.
                                     Result := "Void"
                                                                     What to do: ensure target of the call is attached.
                           end
                           Result := element.out
                                                                     Class: NODE [G -> detachable ANY]
                                                                     Feature: out
                  end
                                                                     Type: Generic #1
end
                                                                     Line: 55
                                                                          end
                                                                        Result := element.out
                                                                        end
```

```
class NODE[G -> detachable ANY] inherit
    ANY redefine out end
    DEBUG_OUTPUT redefine out end
creation
    make ...
feature
    element : detachable G ...
feature -- out
    debug_output: STRING
            -- String representation for debugging
        do
            Result := out
        end
    out: STRING
        do
            if attached element as l_e then
                Result := l_e.out
            else
                Result := "Void"
            end
        end
end
```



```
t1: BOOLEAN
      local
             node: NODE[detachable STRING]
             s: STRING
      do
             comment("t1: First test node")
             create node.make (Void, Void, Void)
             Result := node.element ~ Void
                    and node.previous = Void
                    and node.next = Void
                                                          1 Error 0 Warnings
             check Result end
                                                          Description
                                                          O VUTA(2): Target of the Object_call might be void.
             s := node.previous.element
                                                             Error code: VUTA(2)
      end
                                                             Error: target of the Object_call might be void.
                                                             What to do: ensure target of the call is attached.
                                                             Class: TESTS
                                                             Feature: t1
                                                             Type: detachable NODE [detachable STRING_8]
                                                             Line: 38
                                                                check Result end
                                                             -> s := node.previous.element
                                                               end
```

```
class
       NODE[G -> detachable ANY]
   feature -- queries
       element : detachable G
       previous: detachable NODE[G]
       next: detachable NODE[G]
   feature -- commands
       set_element(e: detachable G)
           do
               element := e
           ensure
               element_changed:
Add a
                   element = e
public
               previous_unchanged:
                   attached (old previous) as old_previous
command
                   implies
                   attached previous as new_previous
                   and then old_previous.element = new_previous.element
           end
```

```
t1: BOOLEAN
    local
        node: NODE[detachable STRING]
        s: STRING
    do
        comment("t1: First_test node")
        create node.make (Void, Void, Void)
        Result := node.element ~ Void
            and node.previous = Void
             and node.next = Void
                                          Cannot do:
        check Result end
                                          node.element := "Yay!"
        node.set element ("Yay!"
        if attached {STRING} node.element as e
        then
            s := e
        end
        Result := s ~ "Yay!"
    end
```