

Extract Slice Refactoring

Ran Ettinger

One Day Workshop in Refactoring Functional Programs

The University of Kent, Canterbury, Kent

February 9th, 2004

Goal: Enhanced Code Reusability

- **Existing code:** you have a function that computes several results and you wish to reuse one of those, in isolation.
- **Refactor:** extract the requested computation into a function whose name explains the purpose of the computation.

Example: Word Count

```
count :: [Char] → (Int, Int, Int)
```

```
count = snd.foldl counter (False, (0, 0, 0))
```

```
counter :: (Bool, (Int, Int, Int)) → Char →  
         (Bool, (Int, Int, Int))
```

```
counter (inword, (nl, nc, nw)) c
```

```
| c == '\n'           = (False, (nl+1, nc+1, nw))
```

```
| c == '\t' || c == ' ' = (False, (nl,   nc+1, nw))
```

```
| not inword          = (True, (nl,   nc+1, nw+1))
```

```
| otherwise           = (inword, (nl,   nc+1, nw))
```

Count Lines – reuse; no isolation

```
countLines :: [Char] → Int
```

```
countLines = first . count
```

```
countChars :: [Char] → Int
```

```
countChars = second . count
```

```
countWords :: [Char] → Int
```

```
countWords = third . count
```

Count Lines – reuse; in isolation

```
countLines' :: [Char] → Int
```

```
countLines' = snd.foldl linesCounter (False, 0)
```

```
linesCounter :: (Bool, Int) → Char → (Bool, Int)
```

```
linesCounter (inword, nl) c
```

```
  | c=='\n'           = (False, nl+1)
```

```
  | c=='\t' || c==' ' = (False, nl)
```

```
  | not (inword)      = (True, nl)
```

```
  | otherwise         = (inword, nl)
```

Count Lines – reuse; in isolation (2)

```
countLines' :: [Char] → Int
```

```
countLines' = foldl linesCounter 0
```

```
linesCounter :: Int → Char → Int
```

```
linesCounter' nl c
```

```
| c=='\n'    = nl+1
```

```
| otherwise = nl
```

Count Words – reuse; in isolation

```
countWords' :: [Char] → Int
```

```
countWords' = snd.foldl wordsCounter  
  (False, 0)
```

```
wordsCounter :: (Bool, Int) → Char → (Bool, Int)
```

```
wordsCounter (inword, nw) c
```

```
| c=='\n' || c=='\t' || c==' ' = (False, nw)
```

```
| not inword                  = (True, nw+1)
```

```
| otherwise                   = (inword, nw)
```

Count Chars – reuse; in isolation

```
countChars' :: [Char] → Int
```

```
countChars' = foldl charsCounter 0
```

```
charsCounter :: Int → Char → Int
```

```
charsCounter nc c = nc+1
```

Count Chars – reuse; in isolation (2)

`countChars ' ' :: [Char] → Int`

`countChars ' ' = length`

Word Count - refactored

```
count' :: [Char] → (Int, Int, Int)
```

```
count' = split3 countLines' countChars'  
        countWords'
```

```
split3 :: (a→b) → (a→c) → (a→d) → a → (b, c, d)
```

```
split3 f g h x = (f x, g x, h x)
```

Extract Slice in monadic code?

```
eval :: (ExcMonad m, StMonad m) =>
```

```
    Term -> m Int
```

```
eval (Con x) = return x
```

```
eval (Div t u) =
```

```
    do x ← eval u
```

```
       y ← eval t
```

```
    tick
```

```
    if y==0
```

```
        then raise "divide by zero"
```

```
        else return (x div y)
```

Exceptions monad

```
evalEx :: Term → Exc Int
```

```
evalEx (Con x) = return x
```

```
evalEx (Div t u) =
```

```
  do x ← evalEx u
```

```
     y ← evalEx t
```

```
     if y==0
```

```
       then raise "divide by zero"
```

```
       else return (x div y)
```

State monad

```
evalSt :: Term → St Int
```

```
evalSt (Con x) = return x
```

```
evalSt (Div t u) =
```

```
  do x ← evalSt u
```

```
     y ← evalSt t
```

```
     tick
```

```
     return (x div y)
```

A larger example

- Tangled flow error checking in a Java compiler:
 - used-before-assigned vars and blank final fields
 - assigned-twice blank finals
 - constructors not filling in blank final fields
 - unreachable stmts
 - missing return stmts
 - various illegal try/catch stmts

References and Acknowledgements

- The word count slicing example is from “Using Program Slicing in Software Maintenance”, Gallagher and Lyle.
- The monadic example is from “Introduction to Functional Programming”, Richard Bird, second edition, Chapter 10.
- The flow error checking example is from the AspectJ compiler (www.aspectj.org): *FlowCheckerPass.java*.
- Thanks to Mathieu Verbaere for his contribution during his MSc project and to our supervisors Oege de Moor and Mike Spivey.
- <http://web.comlab.ox.ac.uk/oucl/research/areas/progtools/projects/nate/nate.html>