MOVE-6: Zen and the A.R.T. of Progress® Coding – Pattern Matching for Better Code

Steven Lichtenberg
Sr. Technologist
Jenark Business Systems

PROGRESS SOFTWARE Exchange 2006

Goals

- **Define A.R.T**
- **Explain the concept of Elegance**
- **Demonstrate tools available**
- **Tips and Tricks**

Agenda

- Define Elegance – why is it important
- Working with existing code
- Tools you may want to consider
- Re-architect and Re-design – Planning for re-use
- We really don't know what will come along

Agenda

Define Elegance – why is it important

- Working with existing code
- Tools you may want to consider
- Re-architect and Re-design – Planning for re-use
- We really don't know what will come along

What is Elegance?

- Design ideas to maximize efficiency
  - Minimize Resources used
    - Hardware and software
    - Human Resources
- Economy
  - Maximize benefit versus cost
- Aesthetics in code can/should be elegant
- Elegance represents good design

What is Elegance?

"Designs that LOOK good will also BE good"

The Tower and the Bridge – David P. Billington

How Does This Relate To Me?

- Simpler/Cleaner code is easier to maintain
- Standard structures allow for easy migration
- Start with the smallest piece of code to perform a specific task
- Write better code for the long term

A Case For Elegance - Advantages

- Easier to read/maintain
- Potential for future problems is greatly reduced
- Architected to account for future use

A Case For Elegance

```
/**** uglycode.p ******/
DEFINE VARIABLE dollarstring AS CHARACTER NO-
  UNDO.
DEFINE VARIABLE decimalpos  AS INTEGER NO-UNDO.
assign
    dollarstring = STRING(account.balance)
    decimalpos   = INDEX(amountstring,".")
    dollarstring = IF length(amountstring) –
        decimalpos = 1 THEN amountstring + "0"
        ELSE IF decimalpos = ? OR
        LENGTH(dollarstring) – decimalpos = 0
        THEN dollarstring + "00"
    dollarstring  = REPLACE(dollarstring,".","").
```

A Case For Elegance

```
/********** bettercode.p **************/
FUNCTION dollarstring RETURNS CHARACTER
(decamount AS DECIMAL):
    RETURN STRING(decamount * 100).
END FUNCTION.  /* dollarstring */
```

Agenda

- Define Elegance – why is it important
- Working with existing code
- Tools you may want to consider
- Re-architect and Re-design – Planning for re-use
- We really don't know what will come along

Working With Existing Code

- Splitting business logic and presentation layers is the first step
- Continue to split procedures as the need arises and it makes sense
- Keep code as flexible as possible
- Re-evaluate procedures each time you make functional changes

General Rules of Thumb

- Deprecated language elements should be removed
- Internal procedures/functions replace multiple include file
- Move duplicate logic to procedure/function
- Look for similar logic patterns

General Rules of Thumb (cont'd)

- Each piece of business logic should exist only once
- Newer language features make this easier.
- Change structure without changing functionality

Agenda

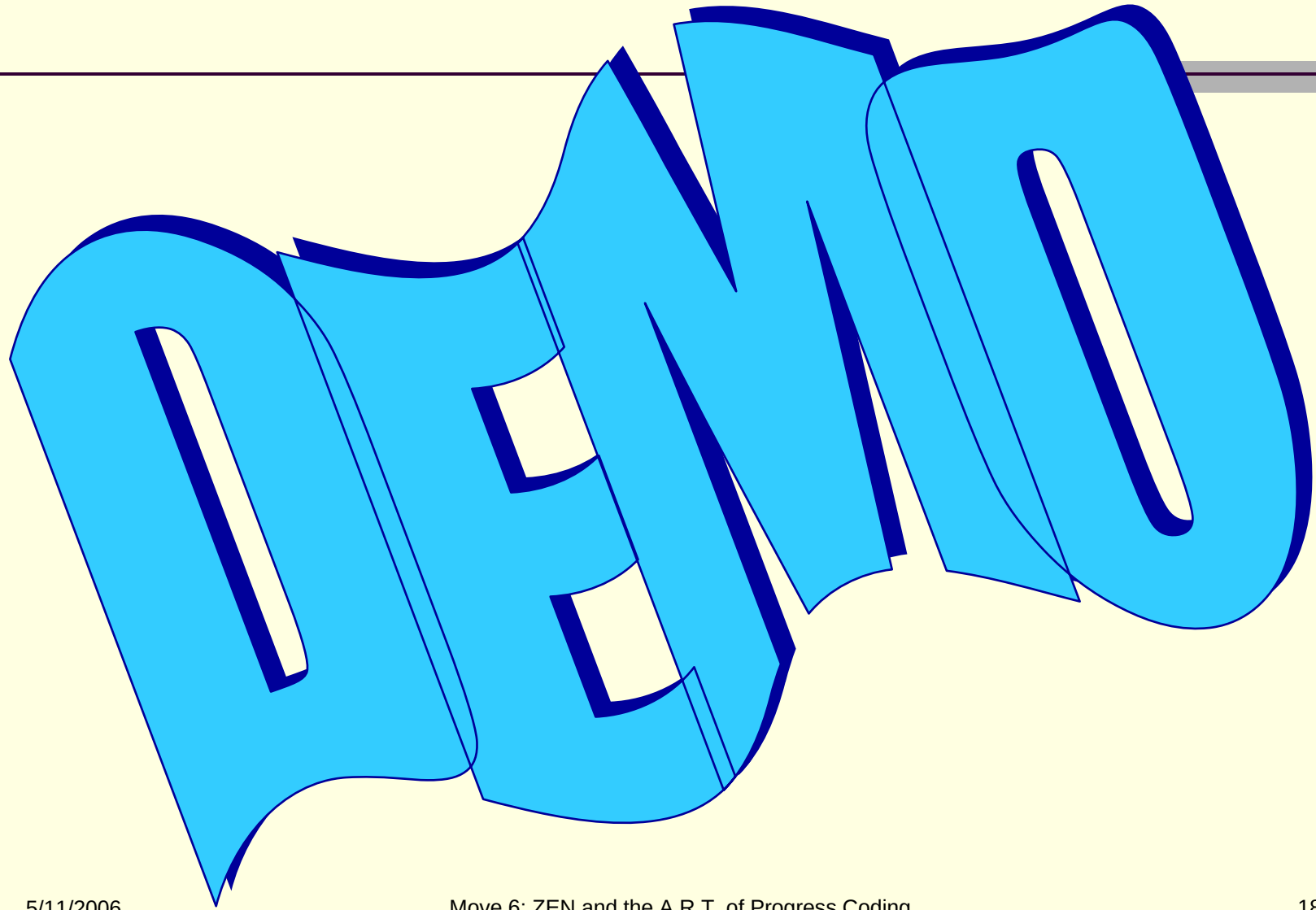
- Define Elegance – why is it important
- Working with existing code
- Tools you may want to consider
- Re-architect and Re-design – Planning for re-use
- We really don't know what will come along

Refactoring Tools - Basic

- Manual Refactoring
- Compile XREF
- Profiler

Refactoring Tools - Advanced

- Proparse – http://www.joanju.com
- Prolint - http://www.prolint.org
- ProRefactor – http://www.prorefactor.org


DEMO

Agenda

- Define Elegance – why is it important
- Working with existing code
- Tools you may want to consider
- Re-architect and Re-design – Planning for re-use
- We really don't know what will come along

Tips And Tricks

- Review code each time you make functional changes
- Continue to split logic as the need arises
- Aim for Single instance of logic
- Try not to take code apart at this point

Architecture Planning

- Plan for change
- Functional changes are separate
- Think in “objects”

Design/Redesign

- Your application will become process based
- Newer delivery methods will be used
- Don’t forget the “old” users

Agenda

- Define Elegance – why is it important
- Working with existing code
- Tools you may want to consider
- Re-architect and Re-design – Planning for re-use
- We really don't know what will come along

Summary

- Consider aesthetics when writing code
  - Pretty code and clean solutions are often better
  - Simpler Code is easier to maintain
- Continue to split code into discrete functional pieces as the need arises and it makes sense
- There are tools available that will make the job easier. Take advantage of them

Summary

- Plan for future changes
  - Time spent now will result in faster implementation later
- Business logic will be needed again.
- Flexibility is the key to success

Further Reading

- **"Who Cares About Elegance? - The Role of Aesthetics in Programming Language Design"**
  - **Bruce J. MacLennan**
  - **http://www.cs.utk.edu/~mclennan/anon-ftp/Elegance.html**
- **"Principled Programming"**
  - **Daniel Read**
  - **http://www.developerdotstar.com/mag/articles/read_princprog.html**
- **Google – "Elegance in Programming"**

Acknowledgements

- John Green/Judy Hoffman – Authors of Proparse/Prorefactor
  - http://www.joanju.com
  - http://www.prorefactor.org
- Jurjen Dijkstra – Author of Prolint
  - http://www.prolint.org
- Progress E-mail Group (PEG)
  - http://www.peg.com
  - Membership: http://www.peg.com/jointoday.html

Questions???????