How Refactoring Helps Bulletproof Your Application

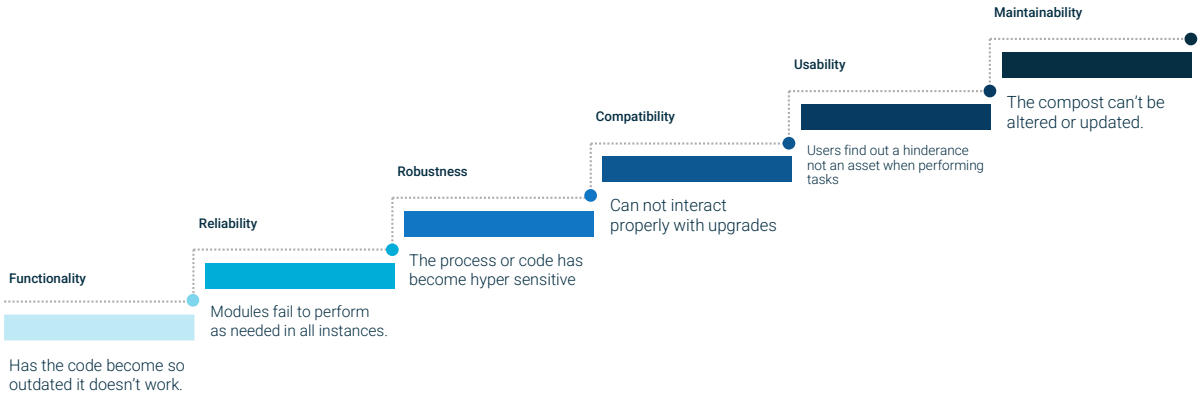
Scott Klement and Yvonne Enselman

What is Refactoring

Restructuring computer code, literally changing the factoring, without changing the external behavior. Intended to improve the software while preserving the functionality.

Code Smell - any characteristic in the source code of a program that may indicate a deeper problem

Issues with these components indicate it is time to refactor



Scenario: Klement's Invoice Inquiry

- They would print invoices in a daily batch process.
- Once printed, you could not reprint an invoice -- you could only look them up on the screen.
- I no longer have access to the code, so I wrote a simpler version to demonstrate

```
                          Display Invoice

   Invoice:  10022       Delivery Address
   Created:  04/06/23    ACME, Inc                      PO: VERBAL-Harry
 Delivered:  04/06/23    Harry Smith                    PO DATE:
  Invoiced:  04/13/23
      Paid:              Anywhere              NY
      Cust:  5250        USA
 Item  T   Qty UOM   Item Description          Price     Weight    LineExtn
70005      19 E     Dell PSeries 27inch Monitor  295.00      9.6    5605.00
70006       9 E     HON Sadie Exec Chair         347.40     25.0    3126.60

                                                                      Bottom
Delivery only available on Mondays, Tuesdays and  Subtotal:          8731.60
Wednesdays during summer. Dock foreman Rob Johns  Ship/Hnd:            69.00
is at phone 414-123-5432                              Tax:           123.00
                                                    Total:    407.4   8923.60
F8=Billing Addr F12=Back
```

Motivation

- User dissatisfaction
- Programmer inability to deliver needed improvements
- Unacceptable time needed for modifications and maintenance
- Incompatible with required upgrades
- Issues with integration with other components on system
- Security concerns
- Outdated skill set needed to work on

What needs to be accomplished

- Improve readability and reduce complexity
- Improve performance
- Determine standardized micro-refactoring
- Possibly adopt automated testing
- Find hidden logic errors or bugs that have been undiscovered

Motivation For Change – Example

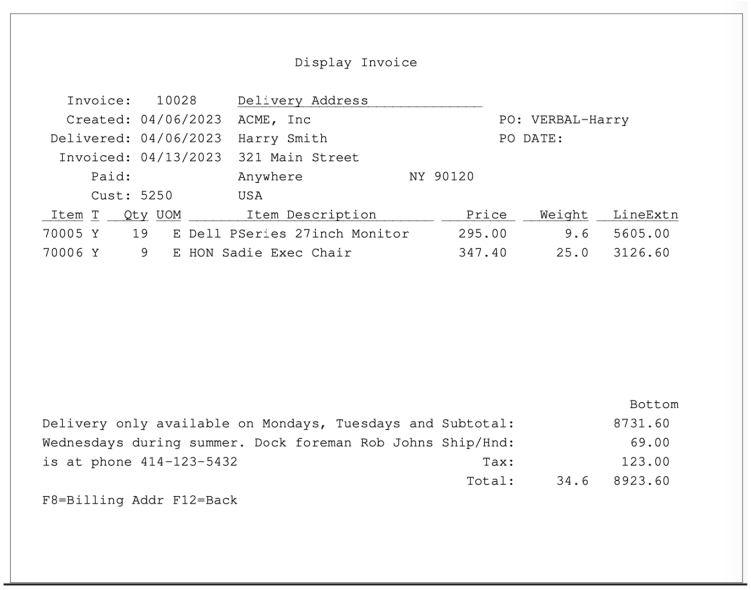
Users would send customer print screens

- Not all information fits on the screen, so they would require multiple print screens.
- Some info is still cut off
- Customer was confused -- trying to piece together the right info from multiple screen shots was difficult.
- Salespeople received complaints, rules were made that accounting could not use this method.

The problem with the example scenario

The Problem

- Users would need to print an invoice (without running the full daily invoice run.)
- They'd do print screens and send to customer
  - But it takes multiple screenshots
  - Some info is cut off.

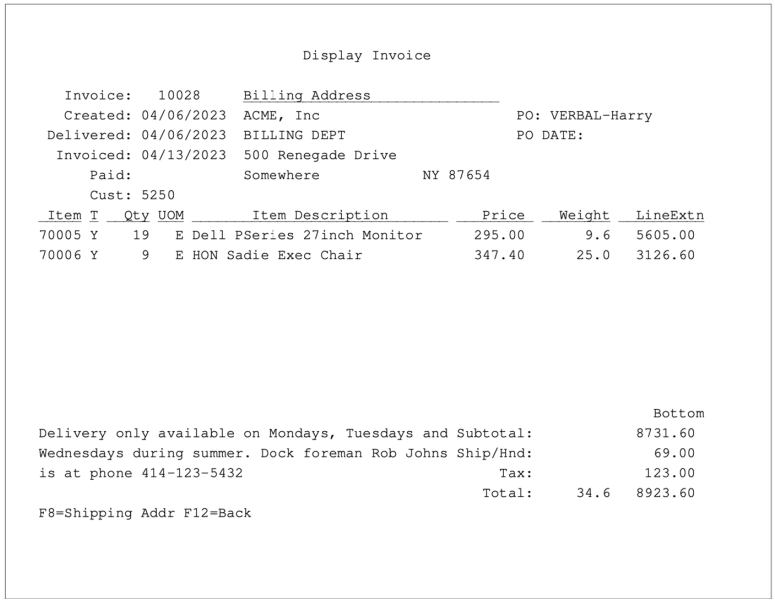


```
                              Display Invoice

   Invoice:   10028     Delivery Address
   Created: 04/06/2023  ACME, Inc                        PO: VERBAL-Harry
 Delivered: 04/06/2023  Harry Smith                      PO DATE:
  Invoiced: 04/13/2023  321 Main Street
      Paid:             Anywhere             NY 90120
      Cust: 5250        USA
 Item T  Qty UOM       Item Description        Price   Weight   LineExtn
70005 Y   19   E Dell PSeries 27inch Monitor    295.00      9.6   5605.00
70006 Y    9   E HON Sadie Exec Chair           347.40     25.0   3126.60




                                                                    Bottom
Delivery only available on Mondays, Tuesdays and Subtotal:          8731.60
Wednesdays during summer. Dock foreman Rob Johns Ship/Hnd:            69.00
is at phone 414-123-5432                                Tax:          123.00
                                                      Total:   34.6 8923.60
F8=Billing Addr F12=Back
```

The problem with the example scenario

The Problem

- Second screen shows billing info
- But notice the end of the message is cut off.
- Some info is repeated
- Sales deemed this "unacceptable to send to customer"
- Accounting would type it onto an invoice form using a typewriter!

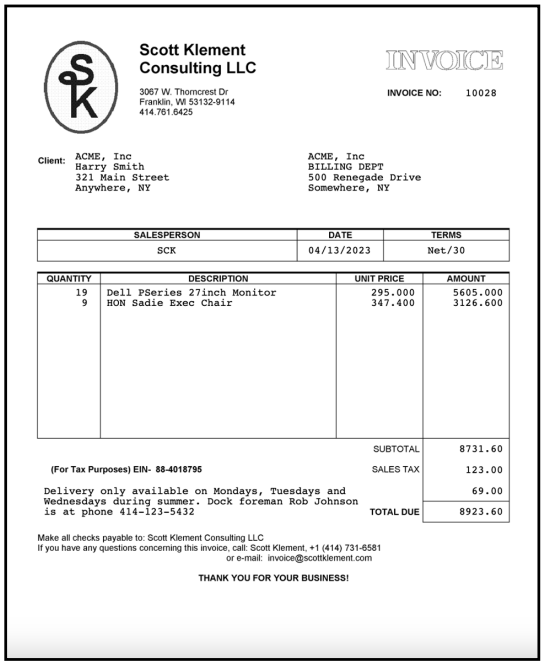


```
                              Display Invoice

   Invoice:   10028     Billing Address
   Created: 04/06/2023  ACME, Inc                        PO: VERBAL-Harry
 Delivered: 04/06/2023  BILLING DEPT                     PO DATE:
  Invoiced: 04/13/2023  500 Renegade Drive
      Paid:             Somewhere            NY 87654
      Cust: 5250
 Item T  Qty UOM       Item Description        Price   Weight   LineExtn
70005 Y   19   E Dell PSeries 27inch Monitor    295.00      9.6   5605.00
70006 Y    9   E HON Sadie Exec Chair           347.40     25.0   3126.60




                                                                    Bottom
Delivery only available on Mondays, Tuesdays and Subtotal:          8731.60
Wednesdays during summer. Dock foreman Rob Johns Ship/Hnd:            69.00
is at phone 414-123-5432                                Tax:          123.00
                                                      Total:   34.6 8923.60
F8=Shipping Addr F12=Back
```

The problem with the example scenario

Solution!

- Once I discovered the problem, I changed the "print screen" to print in an invoice format
- Used:
  - Ability to print with overlay
  - Print to PDF
  - Download via browser



Scott Klement
Consulting LLC

3067 W. Thorncrest Dr
Franklin, WI 53132-9114
414.761.6425

INVOICE

INVOICE NO: 10028

Client: ACME, Inc
Harry Smith
321 Main Street
Anywhere, NY

ACME, Inc
BILLING DEPT
500 Renegade Drive
Somewhere, NY

SALESPERSON	DATE	TERMS
SCK	04/13/2023	Net/30

QUANTITY	DESCRIPTION	UNIT PRICE	AMOUNT
19	Dell PSeries 27inch Monitor	295.000	5605.000
9	HON Sadie Exec Chair	347.400	3126.600

	SUBTOTAL	8731.60
(For Tax Purposes) EIN- 88-4018795	SALES TAX	123.00
Delivery only available on Mondays, Tuesdays and Wednesdays during summer. Dock foreman Rob Johnson is at phone 414-123-5432		69.00
	TOTAL DUE	8923.60

Make all checks payable to: Scott Klement Consulting LLC
If you have any questions concerning this invoice, call: Scott Klement, +1 (414) 731-6581
or e-mail: invoice@scottklement.com

THANK YOU FOR YOUR BUSINESS!

Benefits

- Easier to fix bugs as more readable when troubleshooting
- Organization of monolithic routines to coherent modules
- Moving processes to more applicable classes
- Removing cumbersome or incorrect commenting
- Implementation of design patterns
- Extending the life of a system by bringing into the current standards of the organization

Challenges

- Extraction of system information
- Software structure
    - Data model
    - Intra-application dependencies
    - Team turnover without knowledge capture
- Unclear design decisions made previously
- Architecture of system can be changed
- Updating of HW or OS to use modern features

How Was the New Print Method Solved?

In our example, simply adding the the print screen would be possible, but... all of the logic to calculate the invoice would need to be repeated!

Logic was old and hard to follow. (My rewritten logic is nowhere near as bad -- but there are still benefits.)

In our example, simply adding the the print screen would be possible, but… all of the logic to calculate the invoice would need to be repeated!

Logic was old and hard to follow. (My rewritten logic is nowhere near as bad -- but there are still benefits.)

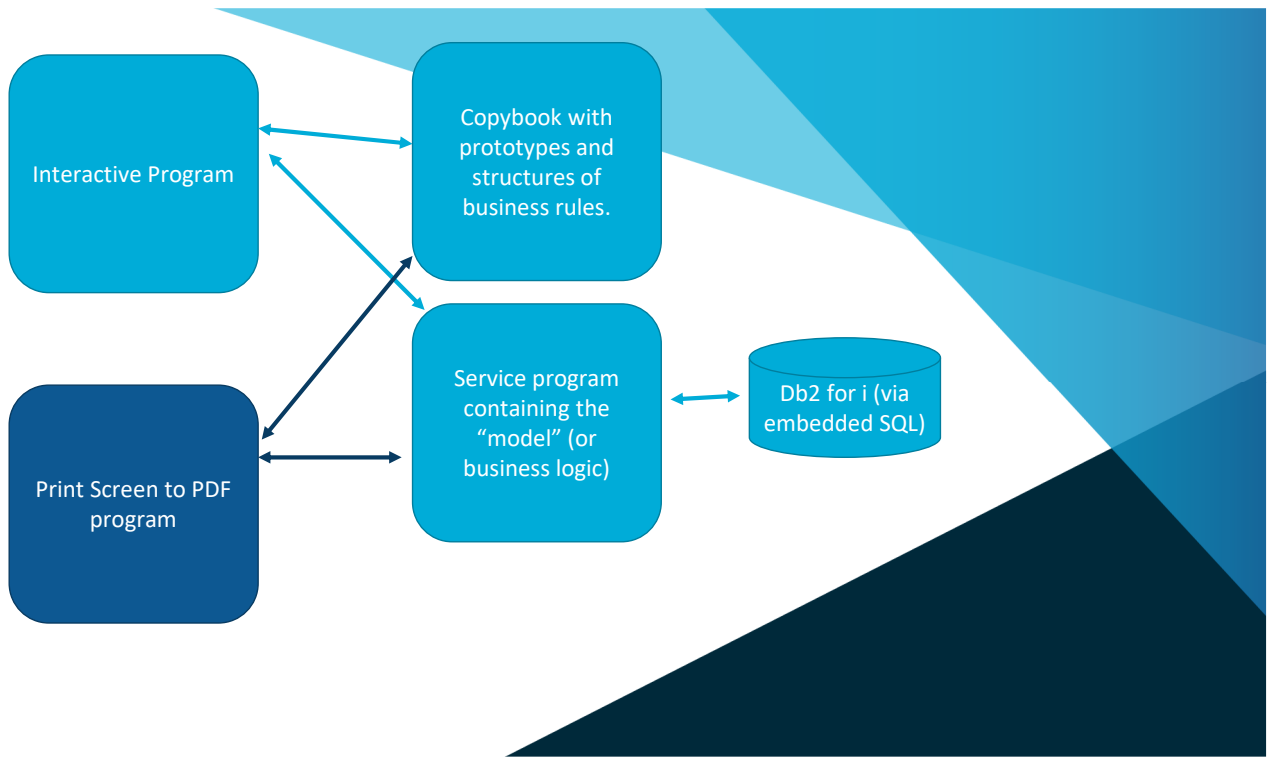
```
     C                   Z-ADD     SHIPPING      SCSHIP
     C                   Z-ADD     TAX           SCTAX

     C     INVNO         SETLL     INVDET
     C     INVNO         READE     INVDET                                 10
     C     *IN10         DOWEQ     *OFF

.....C*0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
     C     ITEMNO        CHAIN     ITEMMAS                                10
     C     IMPRODUCT     IFNE      'Y'
     C                   MOVEL     'N'           IMPRODUCT
     C                   ENDIF
     C   10              MOVEL     *BLANKS       IMPRODUCT

     C                   MOVE      ITEMNO        SCITEMNO
     C                   MOVE      IMPRODUCT     SCPRODUCT
     C                   Z-ADD     QTY           SCQTY
     C                   MOVE      UOM           SCUOM
     C                   MOVEL     DESCR         SCDESCR
     C                   Z-ADD     PRICE         SCPRICE
     C                   Z-ADD     WGTLBS        SCWGTLBS
     C     PRICE         MULT(H)   QTY           SCEXTN

     C                   ADD       SCEXTN        SCSUBTOT
     C                   ADD       SCWGTLBS      SCTOTWGT

     C                   ADD       1             RRN2
     C                   ADD       1             RRN3
     C                   MOVE      *ON           *IN51
     C                   WRITE     INVINQ2S
     C                   WRITE     INVINQ3S

     C     INVNO         READE     INVDET                                 10
     C                   ENDDO
```


- Code is now free-format
- Business logic is separated into a different component
- Logic to load screen is much cleaner/simpler
- Code can be reused from printing program to print GUI invoice

```
err = *blanks;

if invoice_getHeader( inp.INVNO
                    : %date(inp.CRTDATE6: *JOBRUN)
                    : INV) = FAIL;
  err = INVOICE_getLastErr();
  // Report Error To User
endif;


count = invoice_getDetail( INV.INVNO
                         : %date(INV.CRTDATE:*iso)
                         : det
                         : %elem(det));
if count = FAIL;
  ERR = INVOICE_getLastErr();
  // Report error to user
endif;
```

Testing

- How do I know my changes didn't break something?
- How do I code so that I can make changes without breaking something?
  - Well-defined interfaces
  - Proper use of const/value
  - Signatures on service programs
  - Level checks on databases and using SQL or interfaces that will adapt to changes
- Retesting
- Same thing as confirmation testing – only testing the bit you changed, vs retesting the whole system.



Interface – Example

Use external definitions from the database.

Make sure the data structure is defined together with the prototypes in the copybook!

Use CONST, OMIT, NOPASS.

```
dcl-ds INVHDR_t  ext extname('INVHDR')  qualified template end-ds;

dcl-ds INVOICE_HEADER_t qualified inz template;
  INVNO     like(INVHDR_t.INVNO   );
  CRTDATE   like(INVHDR_t.CRTDATE );
  CUSTNO    like(INVHDR_t.CUSTNO  );
  DELDATE   like(INVHDR_t.DELDATE );
  INVDATE   like(INVHDR_t.INVDATE );
  PAIDDATE  like(INVHDR_t.PAIDDATE);
  CUSTPO    like(INVHDR_t.CUSTPO  );
  PODATE    like(INVHDR_t.PODATE  );
  DELNAME   like(INVHDR_t.DELNAME );

.
. ...etc...
.

dcl-pr invoice_getHeader int(10);
  invno   like(INVHDR_t.INVNO)     const;
  crtdate date(*iso)               const options(*omit:*nopass);
  header  likeds(INVOICE_HEADER_t) options(*omit:*nopass);
end-pr;
```

Interface – Example

Caller uses the same DS, same prototypes via the copybook!

Calls existing business logic rather than re-implementing it in each program.

```
/copy invoice_h

.
.

dcl-ds hdr   likeds(invoice_header_t) inz;

.
.

if invoice_getHeader(DSP1.INVNO: *omit: hdr) = FAIL;
  DSP1.MSG = invoice_getLastErr();
  // Handle error
endif;
```

Interface – Example

Caller uses the same DS, same prototypes via the copybook!

Calls existing business logic rather than re-implementing it in each program.

Only export needed routines.

Use the signature to control whether callers do/don't need to be recompiled/bound.

```
dcl-proc invoice_getHeader export;

  dcl-pi *n int(10);
    invno   like(INVHDR_t.invno)     const;
    crtdate date(*iso)               const options(*omit:*nopass);
    header  likeds(INVOICE_header_t) options(*omit:*nopass);
  end-pi;
```

```
strpgmexp signature('INVOICE000000001')
  export symbol(invoice_create)
  export symbol(invoice_getHeader)
  export symbol(invoice_getDetail)
  export symbol(invoice_setHeader)
  export symbol(invoice_setDetail)
  export symbol(invoice_checkItem)
  export symbol(invoice_checkPrice)
  export symbol(invoice_save)
  export symbol(invoice_markPaid)
  export symbol(invoice_delete)
  export symbol(invoice_print)
  .
  .
  export symbol(invoice_getLastErr)
endpgmexp
```

Impact Analysis

- Communicate risk to stakeholders
- What documents and procedures need to be updated or communicated
- What changes need to be made to the codebase
- Impact to the database
- Modernization and complexity factors

Forward compatibility

- Design software that can easily be upgraded to new OS functionality.
- After upgrade is NOT the time to learn that your software no longer works.
- You can't make a change to your software that's needed because it breaks functionality.
- Can't update to new OS because people don't want to change the existing programs – software is too hard to maintain.

The same techniques used for encapsulation also greatly improve impact analysis:

- Code is not repeated
- Changes only in one place
- Test only in one place
- When making updates, we only need to be concerned with exported interfaces.
- CONST lets us know that procedures won't change values.

```
dcl-proc invoice_getHeader export;

  dcl-pi *n int(10);
    invno   like(INVHDR_t.invno)     const;
    crtdate date(*iso)               const options(*omit:*nopass);
    header  likeds(INVOICE_header_t) options(*omit:*nopass);
  end-pi;
```

```
strpgmexp signature('INVOICE000000001')
  export symbol(invoice_create)
  export symbol(invoice_getHeader)
  export symbol(invoice_getDetail)
  export symbol(invoice_setHeader)
  export symbol(invoice_setDetail)
  export symbol(invoice_checkItem)
  export symbol(invoice_checkPrice)
  export symbol(invoice_save)
  export symbol(invoice_markPaid)
  export symbol(invoice_delete)
  export symbol(invoice_print)
  .
  .
  export symbol(invoice_getLastErr)
endpgmexp
```


Thank you!