COMPUTING INTEGRITY
INCORPORATED
60 Belvedere Avenue
Point Richmond, CA 94801-4023
510.233.5400 Sales
510-233.5444 Support
510.233.5446 Facsimile

PROGRESS SOFTWARE BUSINESS ALLIANCE
ISV Partner

Harvesting Editor
19 November 2006

Purpose
When modernizing an application, there is frequently a need to identify and extract portions of the application that contain key business logic so that they can be incorporated into the new version of the application. The Harvesting Editor is intended to facilitate this process by using a rules-based approach to separate what is probably important business logic from the artifacts of the architecture which surrounds it so that a programmer can more easily identify useful code fragments and capture them in a way that they can be re-used. We will use the term "wheat" to identify code which is a good candidate for harvesting and "chaff" for code which we are unlikely to want to harvest.

Context
When companies have an aging legacy application, their options are to continue to try to work with the application as it is, to write a fresh new application, or to attempt to transform the existing application (See http://www.psdn.com/library/entry.jspa?externalID=961&categoryID=58 for a discussion.). Some form of application transformation tends to be the option most likely to produce good results at reduced risk and with controlled cost.

There are several different approaches to application transformation including the formal Application Transformation Approach (ATA) advocated by Progress® (see http://www.progress.com/progress_software/products/services/docs/at_mentoring_ds_final_040105.pdf). Another approach which has recently been discussed in various forums, but without a formal name, is installing an ESB/SOA environment and then progressively converting targeted sections of code into services, a sort of "Getting on the Bus, gradually" approach. Most of these approaches involve a combination of creating new code and harvesting key business logic from existing code. (see http://www.progress.com/progress/ptw/2005/emea/docs/ptw_061.ppt for a discussion in the context of ATA). This current project is aimed at helping the process of harvesting and reusing code from existing applications.

Background
One of the confounding facts in any attempt at architectural modernization is that a great deal of the code in any application is predictable based on the application architecture and the context. Since both the programmer and person paying the programmer are often impressed with the amount of work required to create the code in its current form, this probably deserves some explanation.

Consider the case of the simple file maintenance functions that provide the foundation for any application. Unless there has already been some architectural modernization in the life of the product, chances are that all of these functions within a given application have a great deal of similarity.

Part of these functions is predictable based on other key information, i.e., it derives from the data dictionary and the relationships between tables. Another significant portion of these functions is

© 2006 Computing Integrity Inc. — Proprietary and Confidential, Not for Release or Use Without Permission

simply "the way we write file maintenance programs in the current architecture". It is only what is left when these two aspects are removed that constitutes unique business logic, e.g. validation rules not contained in the dictionary. It is only this portion that is of interest in harvesting.

Thus, if one is migrating to a new architecture and thinks of the same file maintenance functions there, they will also be largely predictable from the data dictionary and those small bits of business logic. In both contexts, the bulk of any program is "chaff" and the wheat is limited to the pieces related to the data dictionary and the small bits of business logic.

Of course, applications don't consist entirely of file maintenance functions, but similar statements apply to inquiry functions and simple list reports. The same principles even apply to many transaction entry functions since those often consist of what amounts to a file maintenance operation coupled with some piece of business logic that creates the impact of the transaction. This means that there is likely to be more business logic than in a simple file maintenance function, but it doesn't necessarily imply that the basic portion that deals with creation, deletion, and editing of records is any different than the file maintenance case.

Harvesting, then, is the process of identifying these pieces of business logic in the midst of all of the other existing code. In some cases, the target can be reasonably apparent because a particular function represents a key transaction in the system, for example. But, in the case of the smaller pieces of business logic which are embedded in large amounts of largely predictable code, it can take considerable effort to identify the desired fragments so that they can be preserved and re-used.

Concept

In the technology of harvesting, there is a continuum between a totally manual review of code at the low end to a hoped for future tool in which all code in an application will be automatically converted to UML, ready for generating a new application. The current project proposes to create a tool which lies between these extremes by using a rule-based structure to assist in determining what is wheat and what is chaff within any given body of code so that the analyst can review the probable wheat and extract it for re-use as appropriate.

It is expected that this goal will be approached in several stages, starting with fairly simple rules which depend only on the code being examined, but later extending to interactions with previously extracted models. In the initial stages, the concept is that we will "gray out" and possibly collapse sections of code which have been evaluated as chaff in order to enable the analyst to review the remaining code more easily and to evaluate it. If it is evaluated as chaff, then the analyst should be able to mark it and collapse it further in order to focus on what remains. In later versions, more sophisticated rules will interact with previously harvested code to determine, for example, whether a particular fragment has already been harvested in another context. Some code fragments, e.g., validation rules, are likely to occur repeatedly in legacy code.

In initial versions, we expect that the actual harvesting will be simply cutting and pasting from the editor into whatever vehicle is going to be used to store prospective logic fragments. In later versions, we would hope to have a more automated process which will directly transfer selected code to a new form, either as a separate code unit or as a component of a model.

While it would be desirable to support all forms of harvesting including cutting and pasting to new .i and .p files, there is a special attraction for supporting harvesting to UML because, in addition to the potential of UML modeling itself, there would be the possibility of predictable relationships between

© 2006 Computing Integrity Inc. — Proprietary and Confidential, Not for Release or Use Without Permission

harvested code and components of the UML model. For example, a field validation would normally be stored as a constraint on an object property, so a field validation in the code could be checked against the object property to determine whether this constraint had already been harvested. To do the same with .i and .p files would require a very artificial naming convention.

Initial Rule Set and Operations

In the initial implementation, the goal will be to classify any one block of code as wheat, chaff, or unknown, where "wheat" is material considered a good candidate for harvesting, "chaff" is material considered to be unlikely to be worth harvesting, and unknown is anything that doesn't fall into either category. Chaff sections will be indicated by a 15% grey background; wheat sections by blue text; and unknown sections by black text. Sections will be boxed or delimited in some way so that one can easily select the whole section. Ctrl-Plus will "promote" a section from chaff to unknown and unknown to wheat; ctrl-Minus will "demote" a section from wheat to unknown and unknown to chaff. Ctrl-[will collapse a marked section to a single indicator line; ctrl-] will undo the collapse to visible text. Ctrl-C can be used to copy an entire section to the clipboard for use in pasting into the desired harvesting repository.

Initial rules to identify "chaff" will be:

1. All DEFINE statements. While the variables defined may be needed for harvested wheat code, that code will typically be packaged differently than it appears in the source program and is likely to be refactored. Thus, one generally won't want to capture the variable definitions with the code because the definitions in the harvested code are likely to change in form. Also variable definitions are often widely separated from their use, making harvesting them both as a unit difficult..
2. All lines consisting only of whitespace. Trailing whitespace will be included in a preceding chaff section; preceding whitespace not already included in a chaff section below will be added to a chaff section which trails it. Alternatively, an option might be provided to simply eliminate any whitespace.
3. All include references, although one can drill down into the include and harvest from it as well. Include references themselves are marked as chaff because it is extremely unlikely that they will be harvested as such.
4. Simple assignments including:
   4.1. Simple assignment of a value from a database table to a local or shared variable.
   4.2. Assignment of literals to a local or shared variable.
5. All UI updates and displays.
6. Access to "system" tables, a user provided list.
7. Comments (see below).

Initial rules to identify "wheat" will be:

1. FORM, DEFINE FRAME, and implicit FORM statements that define UI layout. These are included based on the assumption that one will be trying to capture the general screen layout, e.g., in a fashion similar to Pro/Dox, even though the details of the UI and the technology of its display will be significantly different in the rearchitected code.
2. VALIDATE statements for database fields (might check dictionary and/or a good early candidate for checking previously harvested code per some convention).
3. Flow of control logic. A flow of control block whose contents are entirely UI statements or other chaff will be considered chaff as well.
4. Database access other than to "system" tables.

© 2006 Computing Integrity Inc. — Proprietary and Confidential, Not for Release or Use Without Permission

Note that the simple assignments rule does not include any assignment in which there is computation, since that might be an indicator of a business rule. Some form of the simple assignment might be required to supplement a harvested piece of logic in order to provide appropriate initial values, but these assignments are moderately likely to be of a different form than in the source code. It is also common for them to be physically removed from the place where the value is used. Note also that while the values associated with such variables may well be important in determining control flow, most such control flow will not be harvested in the form it is in. E.g., a file maintenance program might have sections for creating, modifying, or deleting a record depending on whether a record already exists with the specified key and/or some user input. While the code in each section related to what one does to create a record, modify a record, or delete a record may be captured in three code fragments, the control flow leading to those blocks is a part of the local architecture of the old program and will be implemented differently in a new architecture.

While possibly not in the initial implementation, it would be desirable to be able to identify blocks of code such as the following as chaff:

```
do for uom:
  find uom of item no-lock.
  display uom.description[1].
  uom--code = item.uom.
end.
```

Here there is a strongly scoped block, i.e., we know that there are no references to the UoM buffer outside this block that are not inside their own strongly scoped blocks, and within that block there is a no-lock find, a display, and an assignment to a local variable, i.e., it should be a block of chaff.

Also, specific to the code in the samples attached, all blocks referencing init-val and condition should be chaff, but I'm not immediately sure how to make that into a rule. This highlights the need for base rules that are likely to be used with any code and site-specific rules that are added based on the particular body of code currently being harvested.

It could be desirable to eliminate all blank lines, but this would limit any tools for linking back to the original. Instead of attaching them to adjacent chaff blocks as is suggested above, an alternative would be marking them as chaff in their own right and default to displaying these as compressed.

It would probably be useful to "pretty print" to standard indentation prior to marking up the text so that the indentation was an accurate rendering of block structure.

Wheat and chaff rules should have a "weight" and a run-time option should be provided to only mark up wheat or chaff that exceeded a certain weight value. Separate values should be provided for wheat and chaff. E.g., one might assign the comments rule for chaff to -1 and then a cutoff value of 0 would mark comments as chaff and a cutoff of -1 would not. It should also be easy to simply turn a particular rule off when desired. Each chaff rule might also be associated with a flag as to whether initial display should be compressed.

A button or keystroke should be provided to bring in any individual include in the fashion of the COMPILE LIST option. A run-time preference could be provided to default to this behavior or to not bringing in the include.

© 2006 Computing Integrity Inc. — Proprietary and Confidential, Not for Release or Use Without Permission

An alternate treatment of comments would be to mark them as wheat or chaff according to the nature of the node to which they are associated, typically the line of code following for comments that occupy one or more lines.

Implementation

The harvesting editor will be created as an Eclipse plug-in for use in the context of the OpenEdge® Architect or independently. The mechanism for interface to a UML tool is a matter needing further study.

Consideration should be given to implementing the rules as written in an ABL subset or something that looks a great deal like ABL and then compiling these into Java for execution. This would make it considerably easier for a non-Java programmer to extend the editor for his or her own needs.

Later Developments

The long term goal is to advance this technology to the extent that we can achieve automated extractions without the need for operator intervention. When development has advanced sufficiently to begin doing automated extractions, we should probably branch the code development so that there continues to be a harvesting editor available for those who will not use our development technology.

Consideration should be given to enabling rules that will mark unused variables as chaff and which will identify any dead code.

© 2006 Computing Integrity Inc. — Proprietary and Confidential, Not for Release or Use Without Permission

Sample

```
/* po/itemved - Item Vendor Maintenance, main program
 June 85 */
/* ========================================================================== */

define new shared variable selection as character format "x" initial "i".
define new shared variable sel-title as character format "x(23)" extent 3.
define new shared variable tin-entity like po-control.in-entity.
define new shared variable tap-entity like po-control.ap-entity.
define new shared variable po-control-ri as recid.
define new shared variable item-vend-i-ri as recid.
define new shared variable item-vend-t-ri as recid.
define new shared variable item-vend-s-ri as recid.
define new shared variable tprogram as character format "x".
define new shared variable taction as character format "x(10)".

{po/std}

tin-entity = po-control.in-entity.
tap-entity = po-control.ap-entity.
po-control-ri = recid(po-control).

sel-title[1] = "[I]nformation".
sel-title[2] = "[S]pecial description".
sel-title[3] = "[T]echnical description".

repeat:

  display skip(1) space(12)sel-title[1]
          skip space(12) sel-title[2] skip space(12) sel-title[3]
          skip(1) space(12) "Selection -" selection
          with title " " + functdesc + " - Selection " width 50
               centered no-attr-space no-labels 1 down.

  update selection
         validate({val/windex selection its},"ERROR: Invalid selection.")
         help "Enter the desired selection.".

  if selection = "i" then run po/itemved1.
  else if selection = "t" then run po/itemved4.
  else if selection = "s" then run po/itemved5.

end.
```

Comments

Defines

Include

Simple Assignments

Flow of control

Implicit form layout

UI Statement

Flow of control

```
/* po/std - STANDARD DISPLAY FOR PURCHASE ORDER SYSTEM
 June 85 */
/* ========================================================================== */

define new shared variable uom--code like uom.uom-code .
define new shared variable uom--codx like uom.uom-code initial "" .
define new shared variable uom--unit like uom.unit .
define new shared variable uom--exp  like uom.exp-decimal .
define new shared variable uom--max  like uom.max-value .
define new shared variable uom--mask like uom.mask .
define new shared variable uom--desc like uom.description .
define new shared variable uom--dec as decimal label "Qty".
define new shared variable uom--int as integer.
define new shared variable uom--err as logical.
define new shared variable uom--chr as character label "Qty" format "x(10)" .

{ it/lib/shared.i }

find login where terminal-no eq terminalid
            and user-id eq { lib/pro/userid.i } and system eq "po" no-lock.
find entity of login no-lock.
find po-control of entity no-lock.
```

Comments

Defines

Include

“System” table access

© 2006 Computing Integrity Inc. — Proprietary and Confidential, Not for Release or Use Without Permission

```
/* po/itemved1 - Item Vendor Maintenance, information
 June 85 */
/* ================================================================================ */
```

Comments

```
define new shared variable ok as logical.
define new shared variable completed-correctly as logical.
define new shared variable conv-money# like currency.conv-money.
define new shared variable tdesc as character format "x(23)" extent 3.
define new shared variable econ-qty like uom.mask label "Econ-ord-qty".
define new shared variable Landed-cost as character format "x(14)".
define new shared variable uom-desc# as character format "x(6)".
define shared variable item-vend-i-ri as recid.
define shared variable taction as character format "x(10)".
define shared variable tprogram as character format "x".
define shared variable selection as character format "x".
define shared variable sel-title as character format "x(23)" extent 3.
define shared variable tin-entity like po-control.in-entity.
define shared variable tap-entity like po-control.ap-entity.
define shared variable po-control-ri as recid.
define new shared variable twhs-code like in-control.whs-code.
define variable idelete as logical.
define variable tjrnl-no like po-control-d.itm-vend-aud.
define variable tseq like item-vend-a.line-no.
define variable actual-whs like warehouse.whs-code.
define variable twhs-desc like warehouse.description.
define variable default-tax-code like in-tax.tax-code.
define variable tlast-cost like item-whs-d.last-cost.
```

Defines

```
{in/shared}
```

Include

```
find po-control where recid(po-control) = po-control-ri no-lock.
find in-control of po-control no-lock.
```

“System” table access

```
actual-whs = in-control.whs-code.
find warehouse where warehouse.whs-code = in-control.whs-code no-lock.
twhs-code = warehouse.whs-code.
twhs-desc = warehouse.description.
```

Simple Assignments

```
find in-tax of warehouse no-lock no-error.
```

“System” table access

```
if available in-tax then default-tax-code = warehouse.tax-code.
```

Simple Assignments

```
do for po-control-d transaction:
  find po-control-d of po-control exclusive.
  tjrnl-no = po-control-d.itm-vend-aud.
  next-seq-no = next-seq-no + 1.
  tseq = po-control-d.next-seq-no.
end.
```

“System” table access & Simple Assignments

```
tprogram = "I".
```

Simple Assignments

(Continued on next page)

© 2006 Computing Integrity Inc. — Proprietary and Confidential, Not for Release or Use Without Permission

```
main-loop: repeat with no-attr-space frame part-one side-labels row 1 width 77:

  form skip(1) item-vend-i.item-no colon 14 item.description[1] no-label
       uom.description[1] no-label skip item.sort-name colon 14
       item.description[2] no-label
       skip twhs-code colon 14 space(16) twhs-desc no-label
       skip in-tax.tax-code colon 14 space(14) in-tax.tax-desc no-label
       skip item-vend-i.vendor-code colon 14 space(12)
       vendor.name no-label skip
       vendor.sort-name colon 14 /*vendor.city no-label vendor.st no-label*/ skip
       with title " " + functdesc + " - Information " centered.

  clear all.
  display twhs-code twhs-desc.
  if available in-tax then display in-tax.tax-code in-tax.tax-desc.

  prompt-for item-vend-i.item-no
             twhs-code
              validate(twhs-code = "*" or
                 {val/canfind warehouse whs-code twhs-code},
                 "ERROR: the whs-code does not exist in the warehouse file.")
              help "Enter an existing whs-code, or '*' to setup all warehouses".
  assign twhs-code.

  if lookup("item-no",condition) = 0 then
    condition = condition + ",item-no".
  init-val[lookup("item-no",condition) + 1] = input item-vend-i.item-no.
  if lookup("whs-code",condition) = 0 then
    condition = condition + ",whs-code".
  init-val[lookup("whs-code",condition) + 1] = input twhs-code.

  if twhs-code = "*" then actual-whs = in-control.whs-code.
  else actual-whs = twhs-code.

  /* WHS-CODE */
  twhs-desc = "".
  if twhs-code ne "*" then do:
    find warehouse where warehouse.whs-code = twhs-code no-lock.
    twhs-desc = warehouse.description.
  end.
  display twhs-desc.

  /* TAX-CODE */
  if twhs-code ne "*" then
  find in-tax where in-tax.tax-code = warehouse.tax-code no-lock no-error.
  else find in-tax where in-tax.tax-code = default-tax-code no-lock no-error.
  if not available in-tax then do:
    bell.
    message "ERROR: Tax-code not on file.".
    next-prompt twhs-code.
    next main-loop.
  end.
  display in-tax.tax-code in-tax.tax-desc.

  /* ITEM & ITEM-WHS-D */
  find item-whs-d where item-whs-d.in-entity = in-entity
                    and item-whs-d.item-no = input item-vend-i.item-no
                    and item-whs-d.whs-code = actual-whs no-lock no-error.
  if not available item-whs-d then do:
    if twhs-code ne "*" then do:
      bell.
      message "ERROR: Item-warehouse does not exist.".
      next-prompt item-vend-i.item-no with frame part-one.
      next main-loop.
    end.
    tlast-cost = ?.
  end.
  else tlast-cost = item-whs-d.last-cost.
  find item where item.in-entity = in-entity
              and item.item-no = input item-vend-i.item-no no-lock.
  display item.description item.sort-name.
```

Flow of control

Form statement

UI statements

Simple assignment

Should be chaff; no rule yet.

Simple assignment

Comment
Simple assignment
Flow of control
Database access
Simple assignment

UI statements

Comment
Flow of control
Database accessl

UI statements

Comment

Database access

Flow of control

UI statements

Simple assignment

Simple assignment
Database access

UI statements

(Continued on next page)

© 2006 Computing Integrity Inc. — Proprietary and Confidential, Not for Release or Use Without Permission

```
  do for uom:
    find uom of item no-lock.
    display uom.description[1].
    uom--code = item.uom.
  end.
```

Block of chaff

```
  vendor-loop: repeat with frame part-one:
```

Control Flow

```
    output to value("tmp" + terminalid) no-echo.
    output close.
```

Should be chaff; rule?

```
    form item-vend-i.item-vendor colon 15
         item-vend-i.uom-vendor uom-desc# no-label
         item-vend-i.mli colon 64 skip item-vend-i.fob-vendor colon 15
         tdesc[1] no-attr-space no-label item-vend-i.lead-time colon 64
         skip item-vend-i.currency-cod colon 15 tdesc[3] no-label
         econ-qty colon 64 skip
         item-vend-i.conv-uom colon 15
         item-vend-i.duty-code colon 15 tdesc[2] no-label skip
         item-vend-i.taxable colon 15 skip
         item-vend-i.tax-rate[1] label "Tax-rate"  colon 15
         item-vend-i.tax-rate[2] no-label item-vend-i.tax-rate[3] no-label skip
         item-vend-i.misc-pct colon 15 skip landed-cost colon 15
         tlast-cost colon 41
         with no-attr-space frame part-two side-labels title " Details ".
```

Form statement

```
    prompt-for item-vend-i.vendor-code.
```

UI statement

```
    if lookup("vendor-code",condition) = 0 then
      condition = condition + ",vendor-code".
    init-val[lookup("vendor-code",condition) + 1] =
                                      input item-vend-i.vendor-code.
```

Should be chaff; rule?

```
    find vendor
    where vendor.vendor-code = input item-vend-i.vendor-code no-lock.
```

Database access

```
    if not tap-entity matches vendor.valid-entity then do:
      bell.
      message "ERROR: the vendor-code is not allowed for AP-entity" tap-entity.
      next-prompt item-vend-i.vendor-code.
      next vendor-loop.
    end.
```

UI statement block

```
    display vendor.name vendor.sort-name /*vendor.city vendor.st*/.
```

UI statement

```
    find item-vend-i where item-vend-i.in-entity = tin-entity
                     and item-vend-i.item-no = input item-vend-i.item-no
                     and item-vend-i.whs-code = twhs-code
                     and item-vend-i.vendor-code = input item-vend-i.vendor-code
                     exclusive no-error.
```

Database access

```
    /* CREATE ONE LINE */
```

Comment

```
    if not available item-vend-i then do:
```

Control Flow

```
      message "Creating one Item-Vendor...".
```

UI statement

```
      create item-vend-i.
      item-vend-i.in-entity = tin-entity.
      item-vend-i.item-no = item.item-no.
      item-vend-i.whs-code = warehouse.whs-code.
      item-vend-i.vendor-code = vendor.vendor-code.
      item-vend-i.currency-cod = vendor.currency-cod.
      item-vend-i.duty-code = item.duty-code.
      if not vendor.pay-duty then duty-code = "*".
      item-vend-i.tax-rate[1] = in-tax.tax-rate[1].
      item-vend-i.tax-rate[2] = in-tax.tax-rate[2].
      item-vend-i.tax-rate[3] = in-tax.tax-rate[3].
      item-vend-i.conv-uom = 1.
      item-vend-i.uom-vendor = item.uom-code.
      item-vend-i.mli = item.mli.
      do for item-whs:
        find item-whs of item-whs-d no-lock.
        item-vend-i.lead-time = item-whs.lead-time.
        item-vend-i.econ-ord-qty = item-whs.econ-ord-qty.
      end.
```

Table assignments

```
      uom--int = item-vend-i.econ-ord-qty.
```

Simple assignment

(Continued on next page)

© 2006 Computing Integrity Inc. — Proprietary and Confidential, Not for Release or Use Without Permission

```
      { lib/uom/uomdisp.i
          &uom=uom--code
          &int=uom--int
          &chr=uom--chr
          &end=L
          }                                                                  Include
      econ-qty = uom--chr.                                                   Simple assignment
      item-vend-i-ri = recid(item-vend-i).
      run po/itemved2.                                                       Control flow
      display uom-desc# tdesc landed-cost tlast-cost with frame part-two.    UI Statement
    end.

    else do:
      item-vend-i-ri = recid(item-vend-i).                                   Simple assignment
      uom--int = econ-ord-qty.
      { lib/uom/uomdisp.i
          &uom=uom--code
          &int=uom--int                                                      Include
          &chr=uom--chr
          &end=L
          }
      econ-qty = uom--chr.                                                   Simple assignment
      display item-vendor uom-vendor item-vend-i.mli
              fob-vendor item-vend-i.currency-cod
              item-vend-i.lead-time item-vend-i.conv-uom econ-qty             UI Statement
              item-vend-i.duty-code item-vend-i.taxable
              item-vend-i.tax-rate misc-pct tlast-cost
              with frame part-two.
      run po/itemved2.                                                       Control flow
      display uom-desc# tdesc landed-cost with frame part-two.               UI Statement

      /* DELETE ONE LINE */                                                  Comment
      idelete = no.                                                          Simple assignment
      message "Do you want to delete?" update idelete.                       UI Statement
      if idelete then do:                                                    Control flow
        taction = "delete".                                                  Simple assignment
        run po/itemved6.                                                     Control flow
        if opsys <> "MSDOS" then
          input through quoter value("tmp" + terminalid) no-echo.
        else do:
          output to nul.
          dos silent quoter value("<" + "tmp" + terminalid + " >" +
          "tmp" + terminalid + ".q").
          output close.
          input from value("tmp" + terminalid + ".q") no-echo.
        end.
        repeat with no-box width 132:                                        Control flow
          create item-vend-a.
          set line.
          jrnl-no = tjrnl-no.
          line-no = tseq.                                                    Table update
          tseq = tseq + .001.
          record-type = "I".
          item-vend-a.in-entity = tin-entity.
        end.
        input close.
        delete item-vend-i.
        next vendor-loop.                                                    Table update
      end.                                                                   Control flow

      /* MODIFY ONE LINE */                                                  Comment
      else do:
        message "Modifying one Item-Vendor...".                              UI Statement
        taction = "update old".
        run po/itemved6.
      end.                                                                   Control flow
    end.
```

(Continued on next page)

© 2006 Computing Integrity Inc. — Proprietary and Confidential, Not for Release or Use Without Permission

```
    ok = no.                                                                    Simple assignment
    do while not ok with frame part-two                                         Control flow
    on endkey undo vendor-loop, next vendor-loop:
      update item-vendor uom-vendor fob-vendor item-vend-i.currency-cod
             item-vend-i.conv-uom item-vend-i.duty-code taxable                 UI Statement
             tax-rate misc-pct mli lead-time econ-qty
             help "Enter the economical/standard quantity of the item ordered.".

      if lookup("item-vendor",condition) = 0 then                               Should be chaff; rule?
        condition = condition + ",item-vendor".
      init-val[lookup("item-vendor",condition) + 1] = item-vendor.
      run po/itemved2.                                                          Control flow
      display tdesc landed-cost uom-desc#.                                      UI Statement
      if not ok then next-prompt econ-qty.
    end.

    if new item-vend-i then taction = "create".                                 Simple assignment
    else taction = "update new".
    run po/itemved6.                                                            Control flow

    /* CREATE AUDIT-TRAIL */                                                    Comment
    if opsys <> "MSDOS" then
      input through quoter value("tmp" + terminalid) no-echo.
    else do:
      output to nul.
      dos silent quoter value("<" + "tmp" + terminalid + " >" +                 Should be chaff; rule?
      "tmp" + terminalid + ".q").
      output close.
      input from value("tmp" + terminalid + ".q") no-echo.
    end.
    repeat with no-box width 132:                                               Control flow
      create item-vend-a.
      set line.
      jrnl-no = tjrnl-no.                                                       Table Update
      line-no = tseq.
      tseq = tseq + .001.
      record-type = "I".
      item-vend-a.in-entity = tin-entity.
    end.
    input close.
  end.

  do for po-control-d transaction:                                              System Table Update
    find po-control-d of po-control exclusive.
    next-seq-no = next-seq-no + 1.
    tseq = po-control-d.next-seq-no.
  end.
end.
```

© 2006 Computing Integrity Inc. — Proprietary and Confidential, Not for Release or Use Without Permission

Po/itemved1 showing collapsed form

```
main-loop: repeat with no-attr-space frame part-one side-labels row 1 width 77:

  form skip(1) item-vend-i.item-no colon 14 item.description[1] no-label
       uom.description[1] no-label skip item.sort-name colon 14
       item.description[2] no-label
       skip twhs-code colon 14 space(16) twhs-desc no-label
       skip in-tax.tax-code colon 14 space(14) in-tax.tax-desc no-label
       skip item-vend-i.vendor-code colon 14 space(12)
       vendor.name no-label skip
       vendor.sort-name colon 14 /*vendor.city no-label vendor.st no-label*/
skip
       with title " " + functdesc + " - Information " centered.

  if twhs-code ne "*" then do:
    find warehouse where warehouse.whs-code = twhs-code no-lock.

end.

  if twhs-code ne "*" then
  find in-tax where in-tax.tax-code = warehouse.tax-code no-lock no-error.
  else find in-tax where in-tax.tax-code = default-tax-code no-lock no-error.

  find item-whs-d where item-whs-d.in-entity = tin-entity
                    and item-whs-d.item-no = input item-vend-i.item-no
                    and item-whs-d.whs-code = actual-whs no-lock no-error.

  find item where item.in-entity = tin-entity
              and item.item-no = input item-vend-i.item-no no-lock.

  vendor-loop: repeat with frame part-one:

    form item-vend-i.item-vendor colon 15
         item-vend-i.uom-vendor uom-desc# no-label
         item-vend-i.mli colon 64 skip item-vend-i.fob-vendor colon 15
         tdesc[1] no-attr-space no-label item-vend-i.lead-time colon 64
         skip item-vend-i.currency-cod colon 15 tdesc[3] no-label
         econ-qty colon 64 skip
         item-vend-i.conv-uom colon 15
         item-vend-i.duty-code colon 15 tdesc[2] no-label skip
         item-vend-i.taxable colon 15 skip
         item-vend-i.tax-rate[1] label "Tax-rate"  colon 15
         item-vend-i.tax-rate[2] no-label item-vend-i.tax-rate[3] no-label skip
         item-vend-i.misc-pct colon 15 skip landed-cost colon 15
         tlast-cost colon 41
         with no-attr-space frame part-two side-labels title " Details ".

    find vendor
    where vendor.vendor-code = input item-vend-i.vendor-code no-lock.

    find item-vend-i where item-vend-i.in-entity = tin-entity
                       and item-vend-i.item-no = input item-vend-i.item-no
                       and item-vend-i.whs-code = twhs-code
                       and item-vend-i.vendor-code = input item-vend-i.vendor-code
                       exclusive no-error.

    if not available item-vend-i then do:

      create item-vend-i.
      item-vend-i.in-entity = tin-entity.
      item-vend-i.item-no = item.item-no.
      item-vend-i.whs-code = warehouse.whs-code.
      item-vend-i.vendor-code = vendor.vendor-code.
      item-vend-i.currency-cod = vendor.currency-cod.
      item-vend-i.duty-code = item.duty-code.
      if not vendor.pay-duty then duty-code = "*".
      item-vend-i.tax-rate[1] = in-tax.tax-rate[1].
      item-vend-i.tax-rate[2] = in-tax.tax-rate[2].
      item-vend-i.tax-rate[3] = in-tax.tax-rate[3].
      item-vend-i.conv-uom = 1
```

(Continued on next page)

© 2006 Computing Integrity Inc. — Proprietary and Confidential, Not for Release or Use Without Permission

```
      item-vend-i.uom-vendor = item.uom-code.
      item-vend-i.mli = item.mli.
      do for item-whs:
        find item-whs of item-whs-d no-lock.
        item-vend-i.lead-time = item-whs.lead-time.
        item-vend-i.econ-ord-qty = item-whs.econ-ord-qty.
      end.
      run po/itemved2.
    end.
    else do:
      run po/itemved2.
      if idelete then do:
        run po/itemved6.
        if opsys <> "MSDOS" then
          input through quoter value("tmp" + terminalid) no-echo.
        else do:
          output to nul.
          dos silent quoter value("<" + "tmp" + terminalid + " >" +
          "tmp" + terminalid + ".q").
          output close.
          input from value("tmp" + terminalid + ".q") no-echo.
        end.
        repeat with no-box width 132:
          create item-vend-a.
          set line.
          jrnl-no = tjrnl-no.
          line-no = tseq.
          tseq = tseq + .001.
          record-type = "I".
          item-vend-a.in-entity = tin-entity.
        end.
        input close.
        delete item-vend-i.
        next vendor-loop.
      end.
      else do:
        taction = "update old".
        run po/itemved6.
      end.
    end.
    do while not ok with frame part-two
    on endkey undo vendor-loop, next vendor-loop:
      run po/itemved2.
    end.
    run po/itemved6.
    repeat with no-box width 132:
      create item-vend-a.
      set line.
      jrnl-no = tjrnl-no.
      line-no = tseq.
      tseq = tseq + .001.
      record-type = "I".
      item-vend-a.in-entity = tin-entity.
    end.
    input close.
  end.
end.
```

© 2006 Computing Integrity Inc. — Proprietary and Confidential, Not for Release or Use Without Permission