Extending the Life of Software Assets Through Componentization

Kevin Cruz
Product Manager
Relativity Technologies, Inc.

Agenda

- Challenges to any ADM Project
- Componentization and Re-engineering
- Applying the Techniques
- Summary

ADM Challenges

Code quality degrades over time

- Time pressures lead to technical compromises
- Developers rotate through the application
- Preventive maintenance and re-factoring are secondary priorities
- Changes to business policies and product lines often lead to obsolete code

ADM Challenges

Initial design of older systems often lacked "separation"

- Separation of data from processes and interface
- Separation of business logic from technology bound code
- Separation of distinct business functions

Relativity TECHNOLOGIES

Code Re-engineering Techniques

- Removal of dead data and code
- Domain Specialization
- Computation Extraction
- Range Extraction

Removal of Dead Code – Why?

- Unreachable code
- Unused data structures
- A quick way to reduce the size of the application

Removal of Dead Code – How?

- Determine unreachable points via control flow analysis
- Data element and structure usage analysis

Removal of Dead Code - Reality

Automated code removal needs to be configurable to support client coding practices and modernization goals.

- Handling of data structures (remove all elements, only remove structures, etc.)
- Handling of data and procedure division copybooks (COBOL)

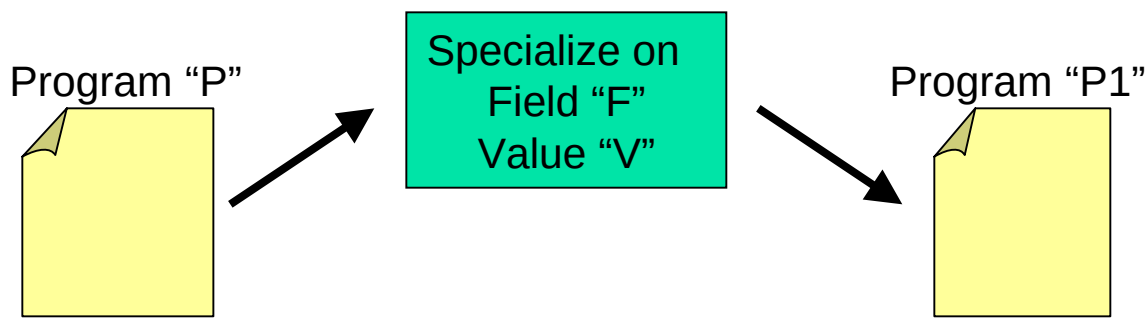
Relativity TECHNOLOGIES

Domain Specialization – Why?

- Changes to business policies or product offerings.
- To understand, identify or classify the distinct threads of business logic
- To disentangle business logic

Domain Specialization – How?

- Computer Science discipline of program specialization
- Derivation of a new program based on some knowledge of the input data.



Domain Specialization – How?

- Requires a deep level of parsing
- Algorithm operates based on user defined variables and values
- Multiple passes allow further specialization to occur based on eliminated data dependencies of previous passes
- Code that becomes unreachable based on specialization is removed.

Domain Specialization – When?

- Need to separate logic based on a limited set of record types
- Eliminate environment specific code
- Remove obsolete code based on data conditions.

Domain Specialization - Reality

Program complexity and "non-standard" coding practices make automating this task imperative.

- Cyclomatic/Essential Complexity well beyond "maintainable" standards
- Internal data flow alter the values being specialized

```
001600
001700 PROCEDURE DIVISION.
001800 PARA-1.
001900     ACCEPT TRAN IN CUSTOMER.
002000     ACCEPT Y.
002100
002200     IF TRAN IN CUSTOMER = '121'
002300        DISPLAY 'XXX'
002400        MOVE 1 TO Z
002500     END-IF.
002600
002700     MOVE '123' TO TRAN IN CUSTOMER.
002800     IF TRAN IN CUSTOMER = '123'
002900        DISPLAY 'YYY'
003000        IF Y = 'A'
003100            PERFORM PARA-2
003200        ELSE
003300            PERFORM PARA-3
003400        END-IF
003500     END-IF.
003600
003700     IF TRAN IN CUSTOMER = '121'
003800        DISPLAY 'XXX'
003900     END-IF.
004000
004100     DISPLAY TRAN IN CUSTOMER.
004200     STOP RUN.
004300
004400 PARA-2.
004500     MOVE '121' TO TRAN IN CUSTOMER.
004600 PARA-3.
004700     IF Z = 1 THEN
004800        MOVE '125' TO TRAN IN CUSTOMER
004900     ELSE
005000        MOVE '126' TO TRAN IN CUSTOMER
005100     END-IF.
```

```
001600
001700 PROCEDURE DIVISION.
001800 PARA-1.
RMWDEL*     ACCEPT TRAN IN CUSTOMER.
002000      ACCEPT Y.
002100
RMWDEL*     IF TRAN IN CUSTOMER = '121'
RMWDEL*     END-IF
002300         DISPLAY 'XXX'
RMWDEL*        MOVE 1 TO Z
002500           .
002600
RMWDEL*     MOVE '123' TO TRAN IN CUSTOMER.
RMWDEL*     IF TRAN IN CUSTOMER = '123'
RMWDEL*     END-IF
002900         DISPLAY 'YYY'
003000         IF Y = 'A'
003100             PERFORM PARA-2
003200         ELSE
003300             PERFORM PARA-3
003400         END-IF.
003600
003700     IF TRAN IN CUSTOMER = '121'
003800        DISPLAY 'XXX'
003900     END-IF.
004000
004100     DISPLAY TRAN IN CUSTOMER.
004200     STOP RUN.
004300
004400 PARA-2.
004500     MOVE '121' TO TRAN IN CUSTOMER.
004600 PARA-3.
RMWDEL*     IF Z = 1
RMWDEL*     ELSE
RMWDEL*        MOVE '126' TO TRAN IN CUSTOMER
RMWDEL*     END-IF
004800         MOVE '125' TO TRAN IN CUSTOMER.
```

Before TRAN = 121 After

Computation Extraction – Why?

- A particular variable that is very relevant to the business is often entangled with a mixture of “less interesting” or non-related code
- Can be used to understand the business rules that calculate the variable
- A “parameterized” component can be extracted

Computation Extraction – How?

- All data and control flow “causes” are included into the analysis.
- When creating a component, all code and data items that do not participate in the calculation are left out.
- A clearly defined interface is created for the component.

Computation Extraction – How?

- Algorithm needs a “blocking” capability to limit included code
- Generic Blocking
  - Block all statements of a specific type.
    (i.e. READ or CICS RECEIVE statements)
  - Block specific statements.
    (i.e. User defines exact statements)

Computation Extraction - Reality

- The analysis of the code associated with a calculation
- Data flow and control flow to specified values can be interrogated interactively

Computation Extraction – When?

- When there is a need to extract or understand the code specific to the determination of a variable
- Appropriate statements for extraction are those where the data is passed out of the program.
- Statements appropriate for blocking are those where data is passed into the program.

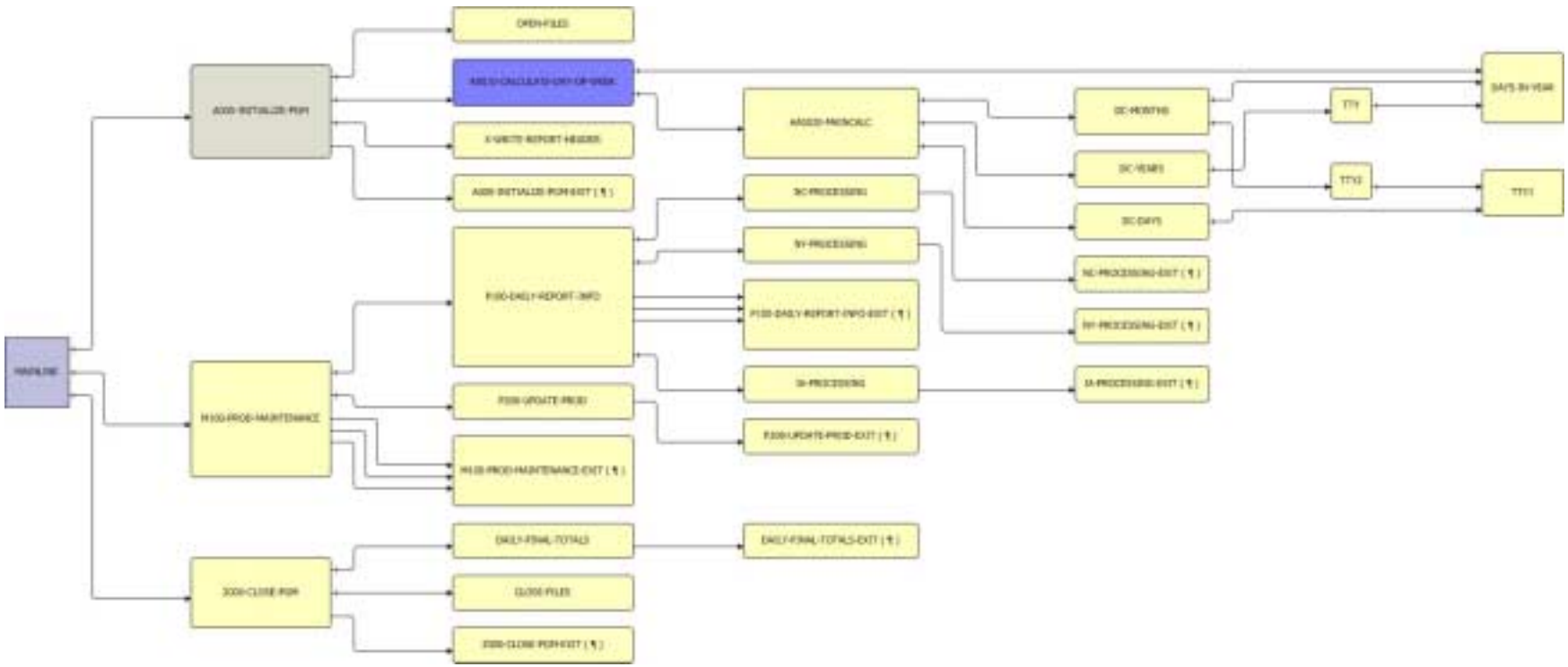
Range Extraction – Why?

- Code exists in system but needs to be re-used or opened up to a service oriented architecture
- In conjunction with duplicate code detection to identify candidates for reuse

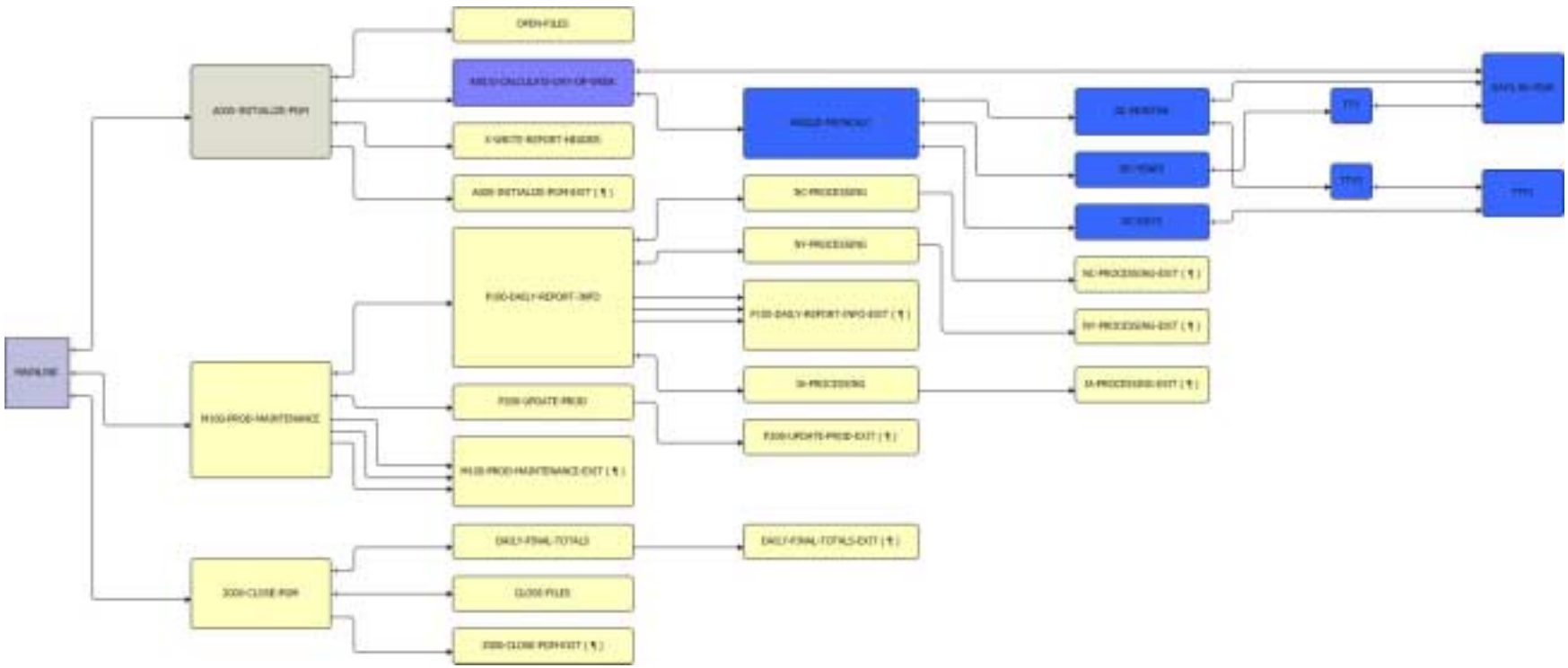
Range Extraction – How?

- Range of code is used as a parameter (seed) for extraction
- New program is created based on the transitive closure of the selected range
- New program can be created as a parameterized component
- Extracted code is removed and a “complement” is created with a call to the new program

Range Extraction – Example



Range Extraction - Closure



Range Extraction – Reality

- Resulting component is structured well and ready for forward engineering
- This technique is more effective with well structured code since it is based on transitive closure
- Works well with code restructuring

Range Extraction – When?

Range Extraction should be used when there is an intention to capture side effects of a calculation

Applying the Techniques

- Techniques can be mixed and applied when appropriate
- Allows modernization to occur iteratively over time, does not necessitate a “big bang” approach
- Complementary to other modernization activities (i.e. code restructuring, clean and shine, etc.)

Summary

- Code degradation over time and business logic tightly coupled with technology based code presents a challenge to any ADM project. The componentization and re-engineering techniques discussed are a means to extend the lifetime of software assets.