



NRC-CNRC

*Institute for
Information
Technology*

Recovering Business Rules from Legacy Source Code for System Modernization

Erik Putrycz, Ph.D.

Anatol W. Kark

Software Engineering Group

National Research Council, Canada

Introduction

- **Legacy software**

```
000009* PROGRAM TITLE:  
000010* DATE CREATED:  JULY 1979  
000011* AUTHOR:         UNKNOWN  
000012* DESCRIPTION:
```

- **Common**

- In 2006, 70% of all transaction systems were written in COBOL

Introduction (2)

- **Modernization**

- Motivations

- High cost to operate legacy system
- Impossible to keep the legacy system up-to-date
- Lack of qualified staff

- New system or integrate legacy with new system

- Need for requirements

- Many requirements buried in the source code

- Recovering business rules major issue

- Recent survey from Software AG: 51% of companies who have difficulties modernizing said that a major issue are “hard-coded and closed business rules”

- **Objective**

- How to recover business rules from legacy source code?

Outline

- **Context**
- **Extraction process**
- **Visualization and navigation**
- **Conclusions**

Outline

- **Context**
- **Extraction process**
- **Visualization and navigation**
- **Conclusions**

Legacy Systems

- They handle and process a large amount of data.
- They have a large user base.
- They involve many business rules linked to legislation, agreements, processes, etc.
- Many business rules are not accurate anymore and require external error and exception processing because of the difficulty to improve and update such system.
- The maintenance of the system often involves changes related to business rules.
- The source code is available – most developments “in-house”.

- Two main classes of the stakeholders are involved with business rules:
 - *Legacy system maintainers*
 - Fix bugs and implement new rules.
 - Need to understand the business rules they are affecting and the execution paths to a specific business rule
 - *Business analysts*
 - Involved in modernization of the legacy system
 - Business rules in legacy system used for validating new requirements or finding requirements
 - Often no background in technologies used in legacy system

Objectives

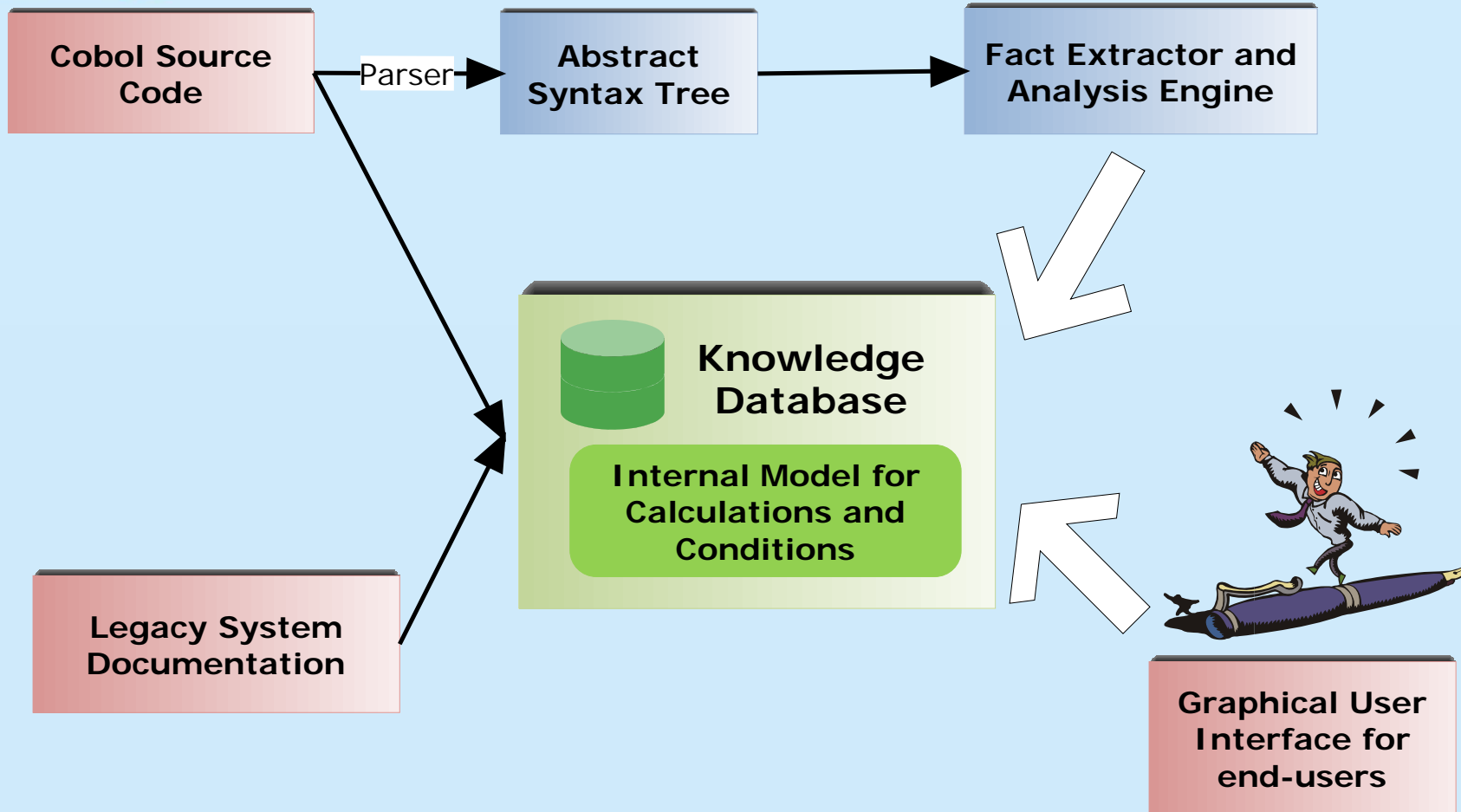
- **Build an extensible and customizable framework for knowledge extraction**
- **Extract business rules in the following form:**
 - If <conditions> then <consequence>
 - <conditions> and <consequence> as easy to understand as possible
- **Use “business terms” instead of programming language constructs**
- **Focus on calculations**
 - Representation of complex calculations in an “easy” format
- **Provide navigation and visualization to locate information**
- **Focus on COBOL legacy software**

Outline

- Context and objectives
- **Extraction process**
- Visualization and navigation
- Conclusions

- **Limited and simple structure**
 - each program = sequence of statements grouped into paragraphs which are executed sequentially
- **Only two forms of branching**
 - one to execute external programs (CALL)
 - another to transfer the control flow to a paragraph (PERFORM). PERFORM is also used for iterations.
- **Each program is associated with a single source file.**
- **Elements of Business Rules**
 - Branching: execute a paragraph
 - Calculations: compute calculations between variables into another
 - Conditions

Platform overview



- **Main elements**
 - Business rules = condition + action
 - Action: calculation or control flow branching
 - Conditions: Boolean operations between identifiers
 - Calculations: arithmetic operations between identifiers
 - Identifiers
- **Keyword indexing all of elements**
- **Source code**
 - All extracted artifacts are linked to source code
- **Variable names translation to business terms**

From Source code to Business Rules

- **Abstract Syntax Tree**
 - Extract machine level representation of the source code
- **Business Rules construction**
 - Locate calculations
 - Locate conditions
 - Combine nested ifs
 - Simplify calculations
 - Simplify conditions
 - Compute dependencies between business rules
- **Integrating documentation and comments**
 - Attach comments to relevant elements

- **Technologies**
 - Object oriented database DB4O for storing and querying business rules
 - Antlr3 (LL-*) parsing for generating AST from COBOL
- **One major legacy system**
 - 630000 lines of COBOL
 - 18 programs
 - 11924 business rules
 - 9292 identifiers
 - 2823 paragraphs
- **Database generation**
 - 4h for generation for extracting all business rules and indexing

Outline

- Context and objectives
- Extraction process
- **Visualization and navigation**
- Conclusions

- **Large amount of data**
 - 18 programs
 - 11924 business rules
- **Business analysts**
 - Main artefacts are requirements
- **Need for locating relevant information to users**
 - System wide navigation
- **Recent studies [Ko and al. 2006]**
 - flexible navigation is a key element for software maintainers

Main User Interface

The screenshot displays the 'Business Rule BR-1208' configuration window. The left pane shows a tree view of various business rules, with 'BR-1208 Work-hrly-rate' selected. The main area is divided into 'Condition' and 'Calculation' sections.

Business Rule BR-1208
NRC.PAH1296J
Calculate and Update for Regular Pay (NRC.PAH1296J)

Condition

- ◆ **Bargaining Unit Designer** in
 - 41006
 - 41007
 - 41008
- ◆ **and**
 - M100BASE BP** equals 9.0
 - and M100BASE BP** not equals 6.0
 - and M100BASE BP** not equals 1.0
 - and M100BASE BP** not equals 3.0
- ◆ **and (Bargaining Unit Designer** not
 - 60405
 - 65405
- or M27ABREV class** not
 - GSMP503**
 - GSMP505**
-)
- ◆ **and Bargaining Unit Designer** not equals 31001

Calculation

$$\text{work Hourly rate} = \frac{2190.0}{\text{work Rate amt BP r}}$$

Ready

Business rules and source code linking

The screenshot displays the Business Rules Navigator application. On the left, a tree view shows a hierarchy of business rules under the 'Pay Program' folder. The rule 'BR-6511 Process-output-reg-detail' is selected. On the right, the source code for 'Business Rule BR-651J' is displayed, titled 'Calculate and Update for Regular Pay (NRC.PAH1296J)'. The code is a COBOL-style program with several conditional blocks and arithmetic operations. Lines 012694 through 012700 are highlighted in yellow.

Business Rules Navigator 0.6.20070614.58

File View

Browse Explore Search

Pay Program
Paragraph
Keyword
Business Rule
Identifier

- BR-6496 R--5de5
- BR-6497 Move-reg-cont-ot-fh-rec3
- BR-6498 R--5de6
- BR-6499 Reg-cont-sub1
- BR-6500 R--5de7
- BR-6501 Write-master-record
- BR-6502 Initialize-department
- BR-6503 Process-input-mast-detail
- BR-6504 Process-input-mast-cntrl
- BR-6505 Fcr-zero-control-fields-f1-f4
- BR-6506 A-d-start
- BR-6507 Address-start
- BR-6508 A-d-start
- BR-6509 Address-start
- BR-6510 Initialize-employee
- BR-6511 Process-output-reg-detail**
- BR-6512 Warning-w2516-pac15-306
- BR-6513 Process-output-mast-detail
- BR-6514 Write-master-record
- BR-6515 Finalize-reg-detail-record
- BR-6516 Read-master
- BR-6517 Initialize-cycle-totals
- BR-6518 Find-pac15-306
- BR-6519 Verify-ia-prl-status
- BR-6520 Initialize-for-pay
- BR-6521 Initialize-for-penadj

Source code for NRC.PAH1296J Business Rule BR-651J

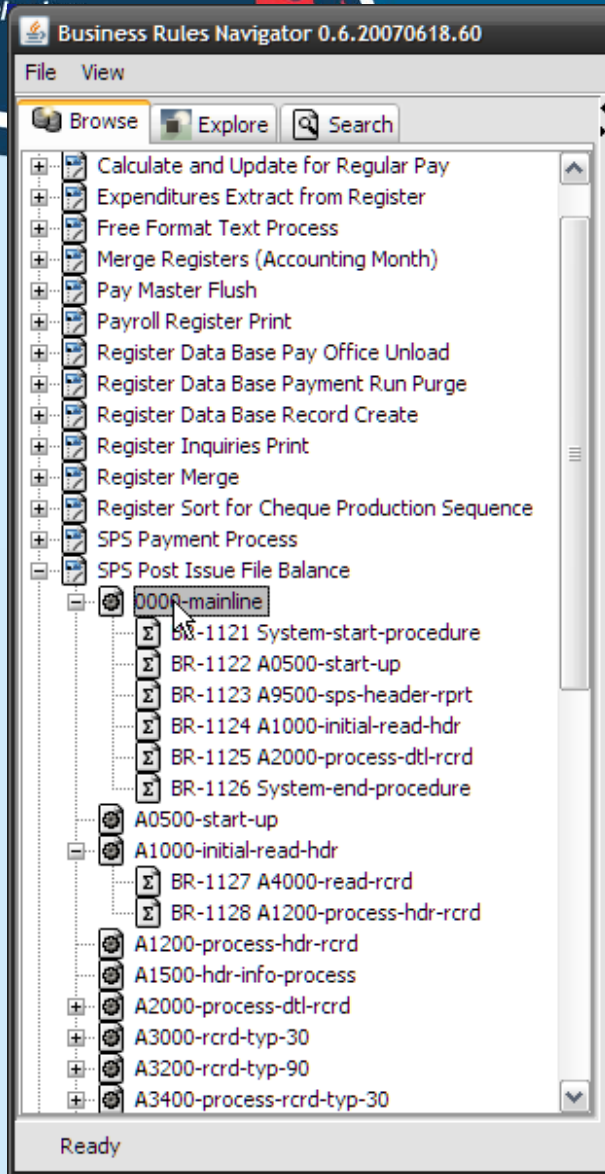
Calculate and Update for Regular Pay (NRC.PAH1296J)

```

012652 PROCESS-INPUT-MAST-DETAIL.
012653 *
012654     IF MID906RECORD-TYPE = 50
012684         IF M943NO-CURR-EARN > 1
012685             COMPUTE A-D-START = (M942NO-PREV-EARN +
012686                                     M943NO-CURR-EARN +
012687                                     M952NO-ADDRESSES) * 2 + 1
012688             COMPUTE ADDRESS-START = (M942NO-PREV-EARN +
012689                                     M943NO-CURR-EARN) + 1
012690         ELSE
012691             COMPUTE A-D-START = (M942NO-PREV-EARN +
012692                                     M952NO-ADDRESSES) * 2 + 1
012693             COMPUTE ADDRESS-START = M942NO-PREV-EARN + 1.
012694 *
012695     IF MID906RECORD-TYPE = 50
012696         PERFORM INITIALIZE-EMPLOYEE
012697     IF EMPLOYEE-GETS-PAID
012698     OR EMPLOYEE-PRCSS-FOR-PENADJ-ONLY
012699     OR EMPLOYEE-PRCSS-FOR-TXBL-BEN
012700     PERFORM PROCESS-OUTPUT-REG-DETAIL.
012701 *
012702     IF EMPLOYEE-GETS-PAID
012703         PERFORM WARNING-W2516-PAC15-306
012704         VARYING A-D-SUB FROM A-D-START BY 1
012705         UNTIL M907BLKS = A-D-SUB - 1.
012706 *
012707     IF FATAL-ERROR-SWITCH = 'OFF'
012708         IF MID906RECORD-TYPE = 50
012709             PERFORM PROCESS-OUTPUT-MAST-DETAIL
012710         ELSE
012711             PERFORM WRITE-MASTER-RECORD.
012712 *
    
```

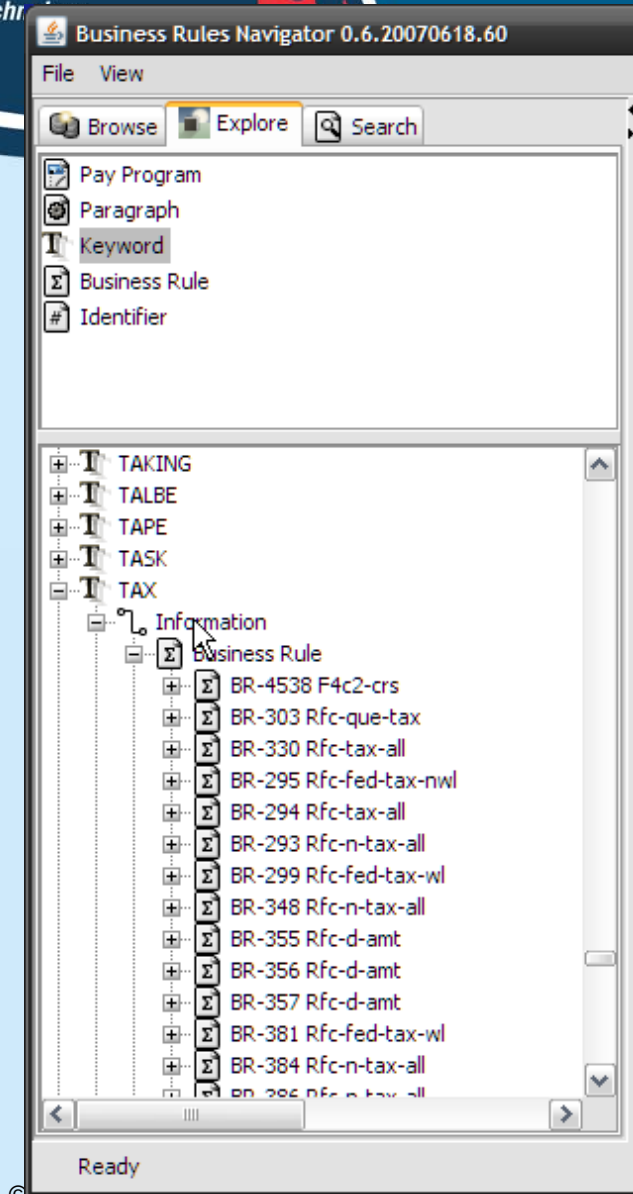
© 2007, Ready

Basic navigation



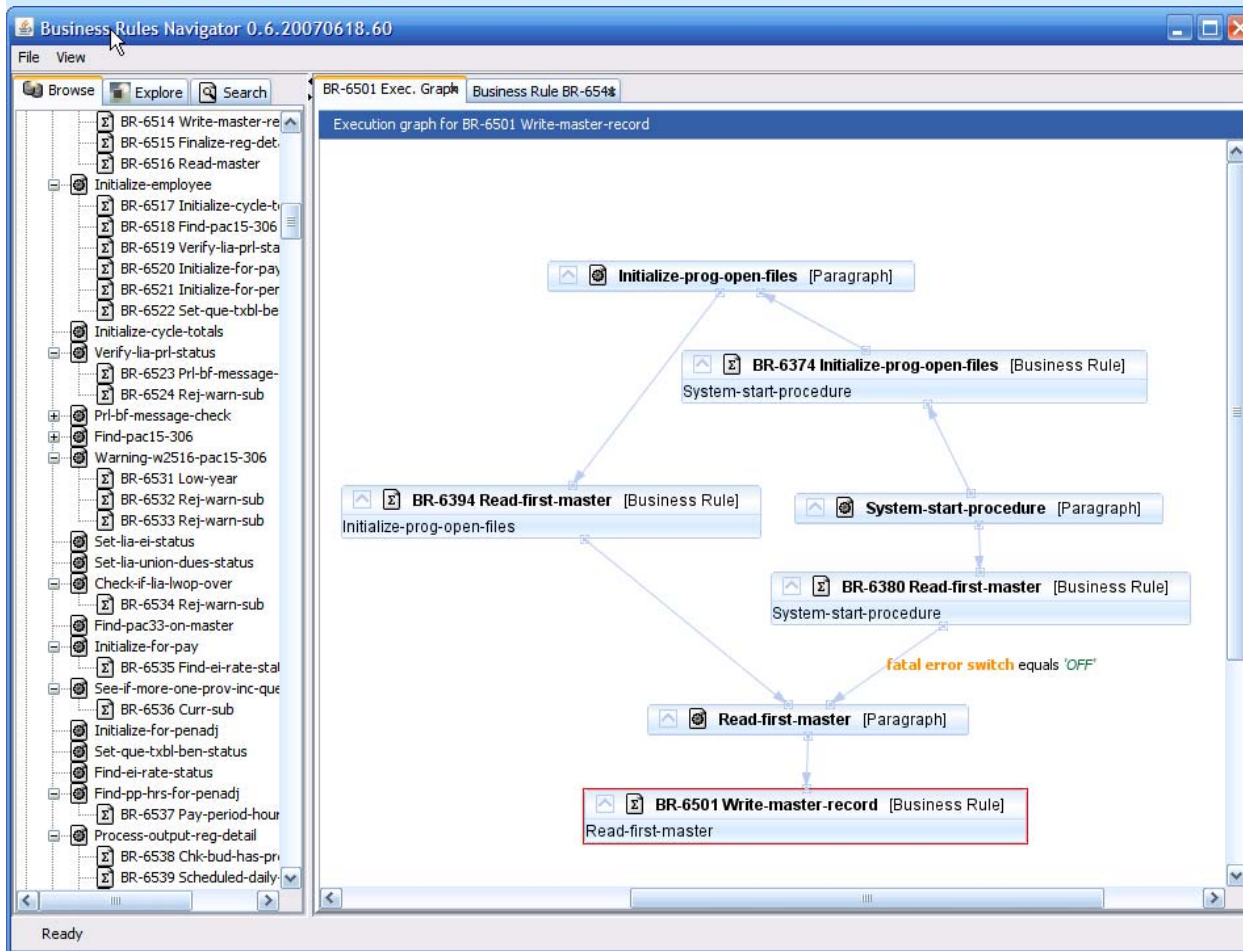
- **Hierarchical navigation**
 - Programs
 - Paragraphs
 - Business Rules

System wide navigation



- **Keyword indexing**
 - Identifiers, comments and COBOL locations are converted to keywords and indexed

Execution path and dependency visualization



- Show all paths leading to the execution of one Business Rule

Outline

- Context and objectives
- Extraction process
- Visualization and navigation
- **Conclusions**

Conclusions

- **Business Rules: major element in legacy software modernization**
 - For system maintainers and business analysts
- **Possible to extract business rules from legacy source code**
- **Novelty**
 - output is targeted at business analysts
 - the business rules translated into non-technical terms.
- **Positive early feedback from business analysts about the value and understandability of the information extracted**

Future

- **Simplified information about branching and loops**
- **Integrate the documentation into the system**
 - Link all elements of the database with existing documentation
- **Data flow analysis**
 - From a generated report of the legacy system, locate all related business rules and calculations
 - State analysis through the execution flow for connecting variables to documented data sources