IEEE Software, July/August, 2004.

Model-Driven Reverse Engineering

Spencer Rugaber, Georgia Institute of Technology
Kurt Stirewalt, Michigan State University

JUNBEOM YOO

Dependable Software Laboratory
KONKUK University

http://dslab.konkuk.ac.kr

2010.08.05

Reverse Engineering

- **Reverse engineering** is the process of comprehending software and producing a model of it at a high abstraction level, suitable for documentation, maintenance, or reengineering.

- From a manager's viewpoint, there are 2 problems
  1. Effort prediction : It's difficult or impossible to predict how much time reverse engineering will require.
  2. Quality evaluation : There are no standards to evaluate the quality of the reverse engineering that the maintenance staff performs.

- Model-driven reverse engineering(MDRE) can overcome these difficulties.
  - Formal specification (SLANG) for describing
    - Domain model
    - Program model
  - Interpretation : annotating connections between two models
  - Automatic code generator : Specware

Adequate Reverse Engineering

- A maintenance manager's uncertainty arises from a lack of understanding about "when the reverse engineering effort is adequate."
  - Adequacy comes from software testing.

- For software testing
  - Adequacy criteria
    - Many coverage criteria
  - Adequacy criteria should be deterministic and measurable.
  - Test suite: the subject of the measurement

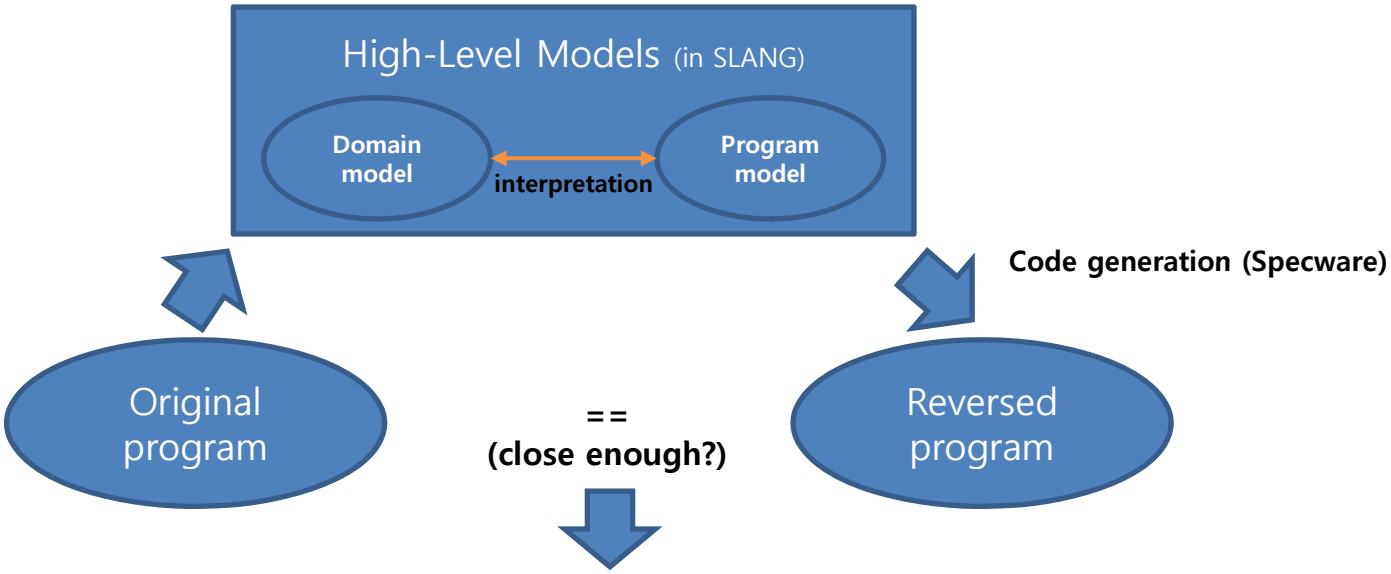
- For MDRE
  - High-level model: the subject of the measurement
  - Adequacy criteria
    - Thoroughness
    - Lucidity

Adequacy Criteria for MDRE

- Thoroughness
  - the extent to which the reverse-engineering effort covers the entire system under examination
- Lucidity
  - the extent to which the reverse engineering sheds light on the system's purpose and how the reversely generated code fulfills that purpose

Reversing Reverse Engineering

- MDRE uses the result of reverse engineering to produce a second version of the original program.



Model-Driven Reverse Engineering

- MDRE uses two models
  - Program model
    - provides a high-level rendering of the functions that the program computes
    - provides a precise statement of the program-computed values but at a higher abstraction level than in the program source code
    - Because algebraic specifications are precise enough to serve as a basis for code generation, they enable measuring the thoroughness of reverse engineering.
  - Application domain model
    - Expresses domain concepts, their relationships, and their meanings independently of a program
    - MDRE makes explicit connections between program constructs and the corresponding domain concepts. (called 'interpretations')
    - useful for assessing lucidity

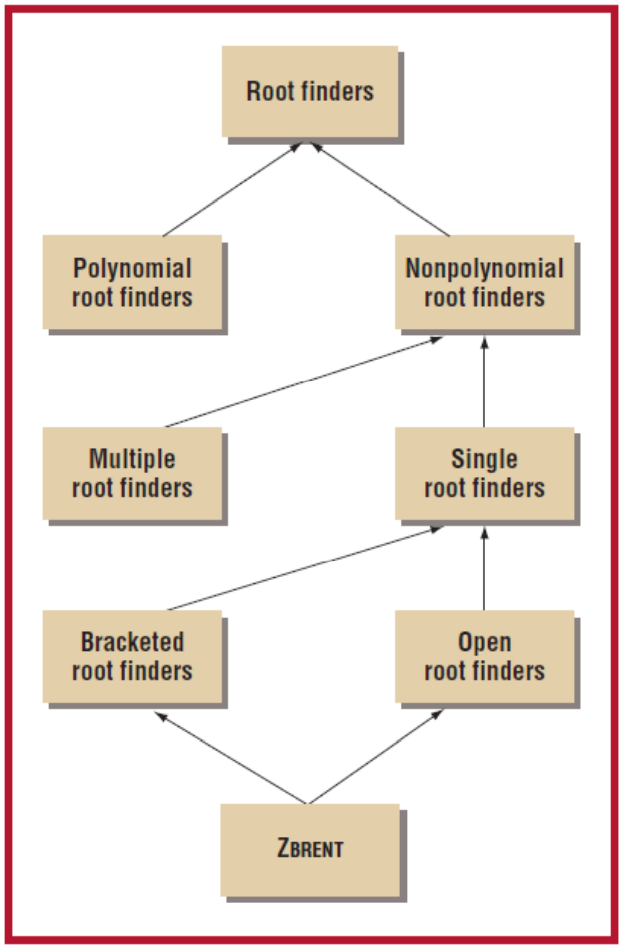
- Both models are described using SLANG
  - Algebraic specification language
  - A part of Specware tool

Example

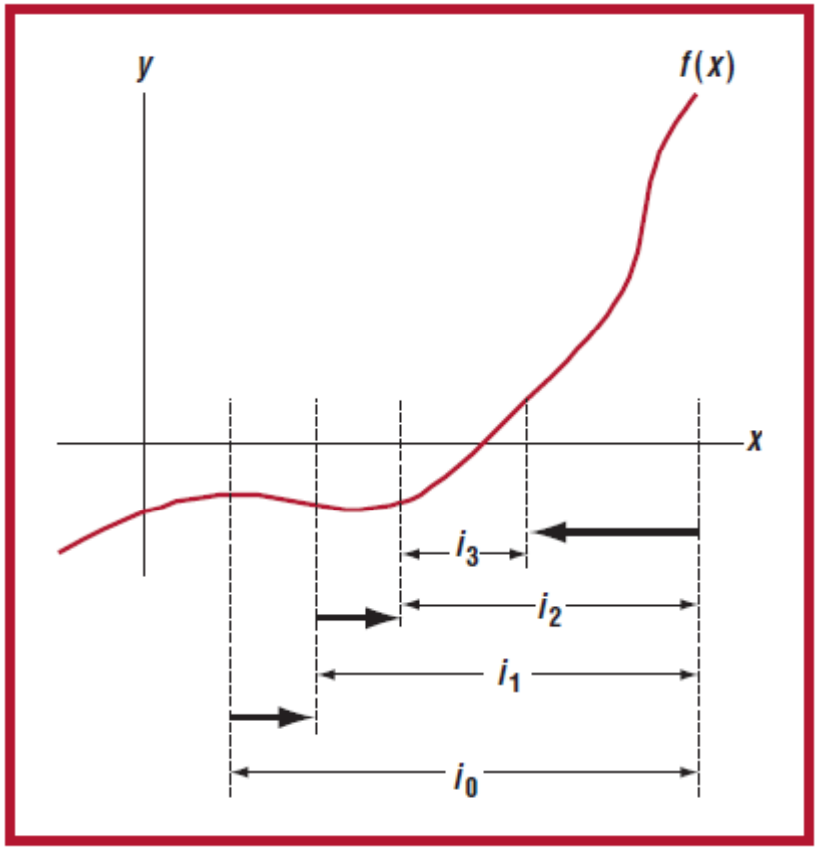
- A numerical application, ZBRENT
  - written in C
  - finding the root of a real-valued function

- Reverse engineering on ZBRENT
  1. Construct a domain model by collecting material from textbooks on numerical analysis
  2. Use SLANG to model both the domain and the program
  3. Use the SLANG code generator in Specware to produce an executable version from our model of ZBRENT,
  4. Compare the executable version with the original program on a set of test functions

Root Finding



- **Refining interval**
- **Stopping criterion**



Iterative interval shrinkage

ZBRENT – SLANG Specification

- Why used ZBRENT
  - It offers several stopping-criteria choices.
  - It features three interval shrinkage methods: bisection, secant, and inverse quadratic interpolation.
  - Victor Basili and Harlan Mills used a variant of ZBRENT in an influential case study. Thus, we can more closely compare our work with theirs.

```
(1)    spec INTERVAL is
(2)      import EXTENDED-REAL
(3)      sort Interval
(4)      sort-axiom Interval = Real, Real
(5)
(6)      op mid-point : Interval -> Real
(7)      definition of mid-point is
(8)        axiom mid-point(a, b) =
(9)          half(plus(a, b))
(10)     end-definition
(11)
(12)     op make-interval : Real, Real ->
(13)       Interval
(14)     definition of make-interval is
(15)       axiom make-interval(a, b) = (a, b)
(16)     end-definition
(17)
(18)     constructors { make-interval }
(19)       construct Interval
(20)   end-spec
```

SLANG INTERVAL specification

SLANG Support for Adequate Models - Morphism

- Specifications in Specware are actual data values that high-level operators called 'morphisms' can manipulate.
  - *Import* includes one specification inside another.
  - *Translate* renames a specification's sorts and operations.
  - *Colimit* combines specifications in a structured way.

- By writing simple specifications and then using morphisms to connect and compose them, developers can cleanly model complex systems.

Interpretation

- Specware uses 'interpretation' to formalize design refinements.
  - Refinements relate abstract domain-model concepts to executable code.

- Operationally, an interpretation demonstrates how Specware implements sorts and operations in one model using sorts and operations in another model at a lower abstraction level.
  - Let reverse engineers directly relate application domain concepts to program constructs

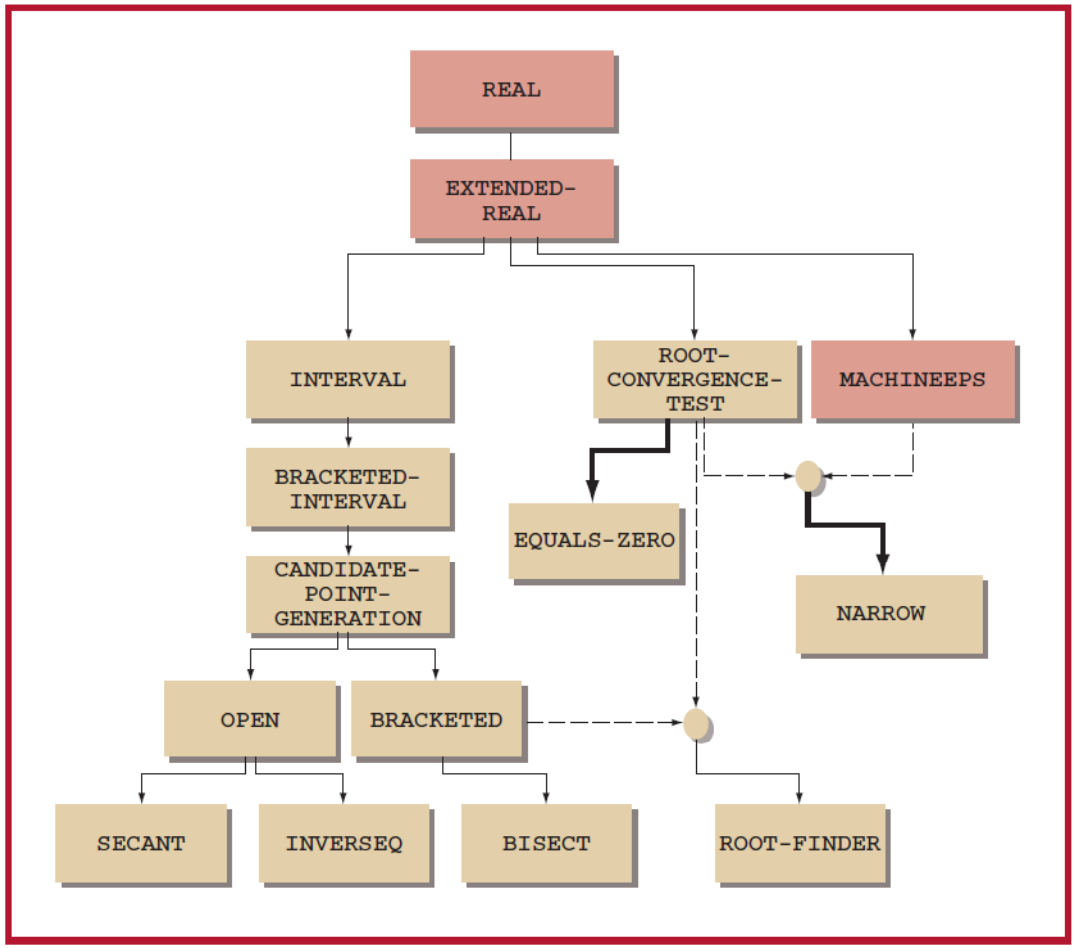
- MDRE assesses lucidity by requiring interpretations to connect domains and implementations.

MDRE process of ZBRENT

1. Construct a domain model by reading descriptions in books and articles on root finding and articulating them in SLANG
   - The domain model provides expectations for concepts that root-finding programs might realize.
2. Construct a program model by expressing the ZBRENT source code as a specification comprising a set of SLANG operation definitions
3. Define SLANG interpretations using an iterative process to connect the program model operations to domain concepts
4. After making a set of connections, execute the Specware code generator, producing an approximation to ZBRENT
   - If the generated program produced results identical to the original, the reverse engineering was thorough.
   - If domain concepts connected to all the program constructs, the reverse engineering was lucid.

- Testing equivalent

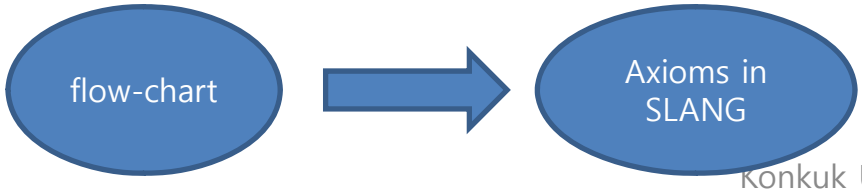
The Root-Finding Domain Model

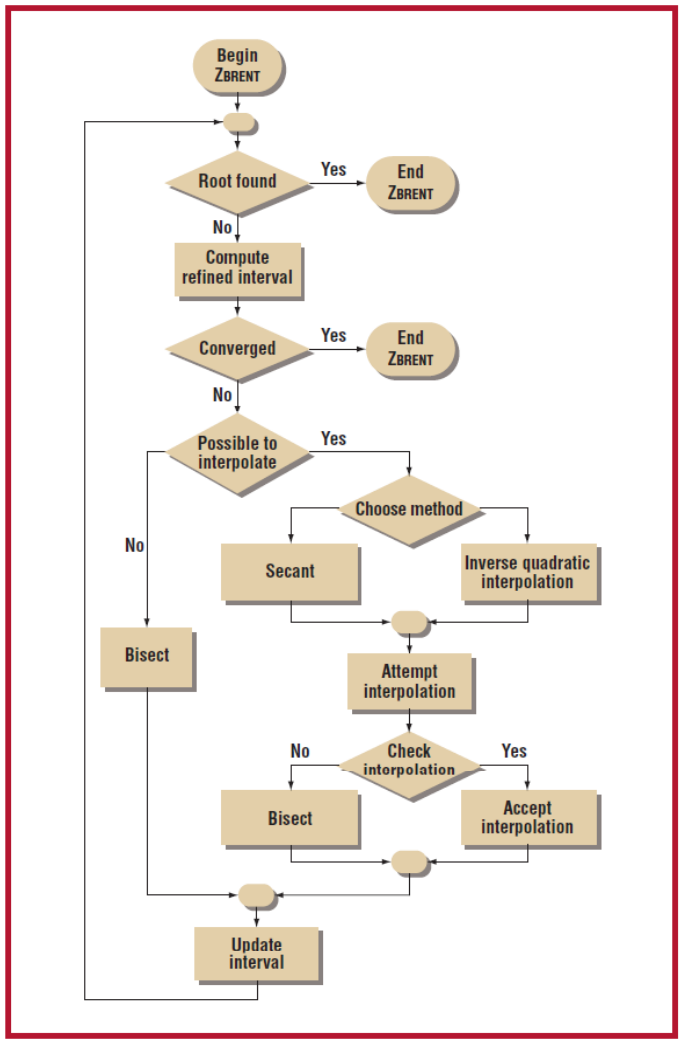


The ZBRENT Program Model

- The ZBRENT algorithm's SLANG specification implements the flow-chart boxes with axioms.

- To construct, reverse engineers perform activities:
  - Define operations corresponding to the various computations performed
  - Model conditional statements using built-in SLANG constructs
  - Use recursion to model iterative computations.
  - Model assignments by passing the resulting state to subsequent operations.

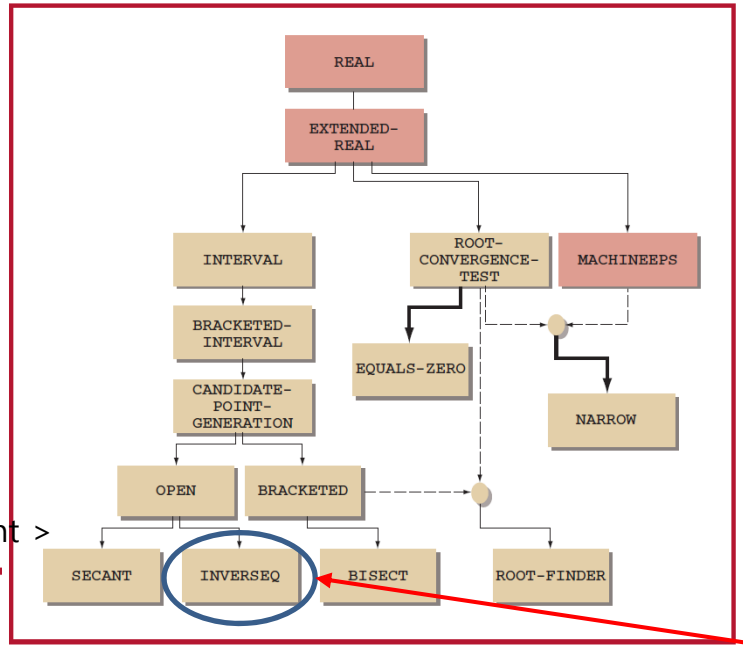




Abstract flow-chart for ZBRENT

Interpretation

< Domain model >



< (a) Original source code fragment >

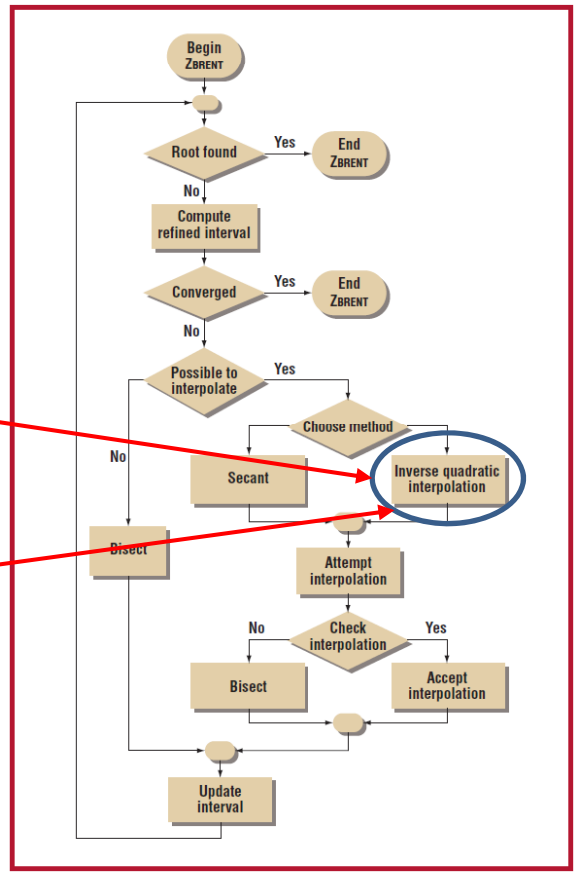
```
(1) q = fa / fc;
(2) r = fb / fc;
(3) q = (q - 1.0) * (r - 1.0) * (s - 1.0);
```

(a)

```
op q : Real, Real, Real, Real -> Real
   definition of q is
      axiom q(s, fa1, fb1, fc1) =
            times(minus(div(fa1,
                  pToNZReal(fc1)),one),
            times(minus(r(fb1, fc1),
                  one), minus(s, one)))
   end-definition
```

(b)

< (b) SLANG operation definition >



< Program model >

Applying MDRE

- Adequacy standards for reverse-engineering efforts would allow maintenance managers to use experience data to predict the cost of such efforts. (Effort prediction)

- An adequacy standard would allow direct comparisons—for example, indicating that one tool provides a more thorough description than another. (Quality evaluation)

Concluding Remarks

- Two prerequisites for using MDRE
  - Mature domain
  - Code generator

- The Unified Modeling Language makes it possible to apply MDRE to a broader class of problems using alternative tool support
  - SLANG → OCL
  - Specware → UML All Purpose Transformer