Software Re-Engineering

Lecture: 08

Dr. Syed Muazzam Ali Shah

Department of Software Engineering

National University of Computer & Emerging Sciences

muazzam.ali@nu.edu.pk

Sequence [**Todays Agenda**]

Content of Lecture

Source Code Reengineering Reference Model (SCORE/RM)

Source Code Reengineering Reference Model (SCORE/RM)

- The SCORE/RM model was proposed by Colbrook, Smythe and Darlison.
- The framework, depicted in Figure, consists of four kinds of elements:
  - function,
  - documentation,
  - repository database, and
  - metrication.

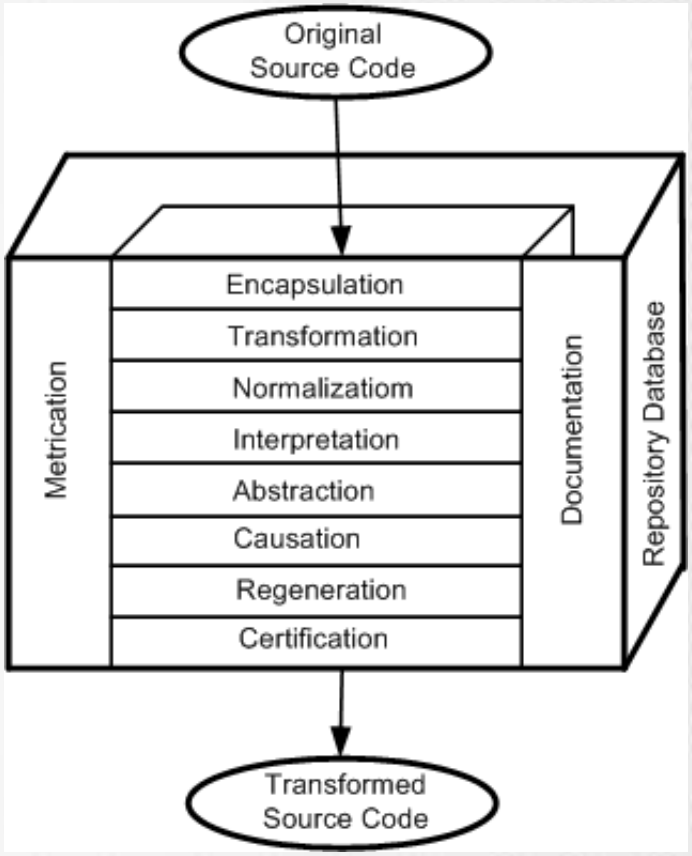

Figure 1 Source code reengineering reference model © IEEE, 1990

Source Code Reengineering Reference Model (SCORE/RM)

- The function element is divided into eight layers, namely:
  - Encapsulation,
  - Transformation,
  - Normalization,
  - Interpretation,
  - Abstraction,
  - Causation,
  - Regeneration, and
  - Certification.

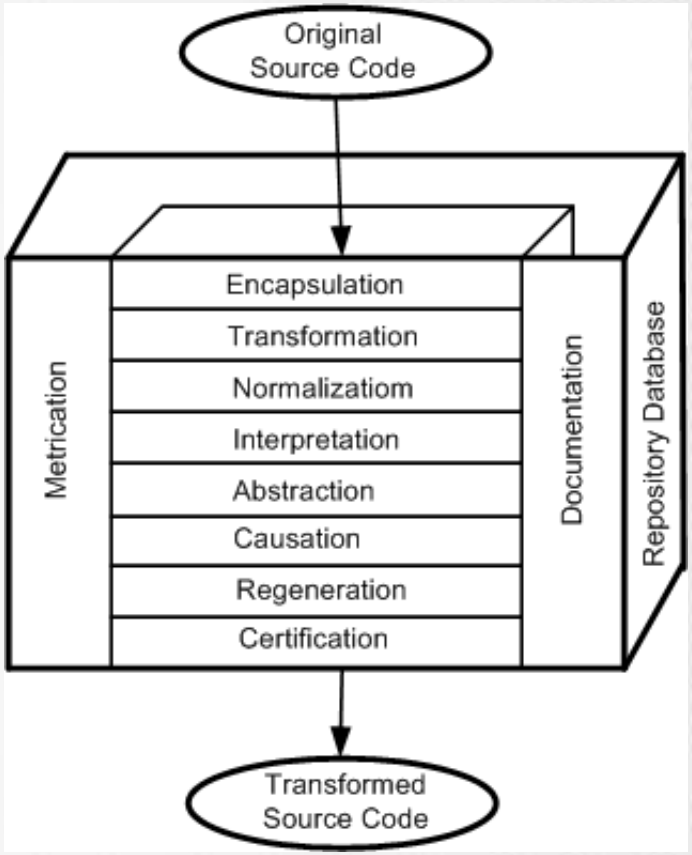

Figure 1 Source code reengineering reference model © IEEE, 1990

Source Code Reengineering Reference Model (SCORE/RM)

- The eight layers provide a detailed approach to:
  - Rationalizing the system to be reengineered by removing redundant data and altering the control flow
  - Comprehending the software's requirements, and
  - Reconstructing the software according to established practices

Source Code Reengineering Reference Model (SCORE/RM)

- Metrication Element:
  - Improvements in the software as a result of reengineering are, quantified by means of the metrication element.
  - The metrication element is described in terms of the relevant software metrics before executing a layer and after executing the same layer.
- Function Element:
  - The top six of the eight layers shown in figure constitute a process for reverse engineering, and the bottom three layers constitute a process for forward engineering.
  - Both the processes include causation, because it represents the derivation of the requirements specification for the software system.

Source Code Reengineering Reference Model (SCORE/RM)

- Documentation Element:
  - The specification, constraints, and implementation details of both the old and the new versions of the software are described in the documentation element.
- Repository Element:
  - The repository database is the information store for the entire reengineering process, containing information i.e. metrication, documentation, and both the old and the new source code.

Source Code Reengineering Reference Model (SCORE/RM)

- The interfaces among the elements are shown in Figure.
- For simplicity, any layer is referred to as (N)– layer, while its next lower and next higher layers are referred to as (N – 1)– layer and the (N + 1)– layer, respectively.

Source Code Reengineering Reference Model (SCORE/RM)

- The three types of interfaces are explained as follows:
  - **Metrication/Function: (N)-MF** – the structures describing the metrics and their values.
  - **Documentation/Function: (N)-DF** – the structures describing the documentation.
  - **Function/Function: (N)-FF** – the representation structures for source code passed between the layers

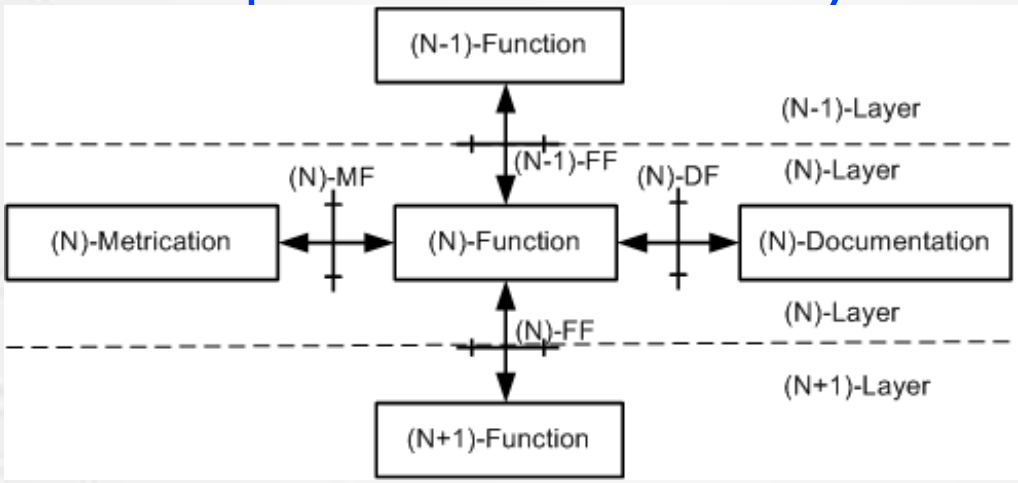

Figure 2 The interface nomenclature © IEEE, 1990

Source Code Reengineering Reference Model (SCORE/RM)

- Let us discuss the function layers.
- Encapsulation.
  - In this layer, a reference baseline is created from the original source code.
  - The goal of the reference baseline is to uniquely identify a version of a software and to facilitate its reengineering.
    - **Configuration Management:**
      - *The changes to the software undergoing maintenance are recorded by following a well-documented and defined procedure for later use in the new source code.*
      - *This step requires strong support from upper management by allocating resources.*

Source Code Reengineering Reference Model (SCORE/RM)

- Let us discuss the function layers.
  - **Analysis:**
    - *The portions of the software requiring reengineering are evaluated. In addition, cost models for the tangible benefits are put in place.*
  - **Parsing:**
    - *The source code of the system to be reengineered is translated into an intermediate language (IL).*
    - *The IL can have serval dialects, depending upon the relationship between the languages for the new code to the original code.*
    - *All the reengineering algorithms act upon the IL representation of the source code.*

Source Code Reengineering Reference Model (SCORE/RM)

- **Test generation:**
  - *This refers to the designing of certification tests and their results for the original source code.*
  - *Certification tests are basically acceptance tests to be used as baseline tests.*
  - *The correctness of the newly derived software will be evaluated by means of the baseline tests.*

Source Code Reengineering Reference Model (SCORE/RM)

- Transformation.
  - To make the code structured, its control flow is changed. This layer performs the flowing functions
    - **Rationalization of control flow:**
      - *The control flow is altered to make code structured.*
    - **Isolation:**
      - *All the external interfaces and referenced software are identified.*
    - **Procedural granularity:**
      - *This refers to the sizing of the procedures, by using the ideas of high cohesion and low coupling.*

Source Code Reengineering Reference Model (SCORE/RM)

- Normalization.
  - In this stage data and their structures are scrutinized by means of the following functions:
    - **Data reduction:**
      - *Duplicate data are eliminated. To be consistent with the requirements of the program, databases are modified*
    - **Data representation:**
      - *The life histories of the data entities are now generated.*
      - *The life histories describe how data are changed and reveal which control structures act on the data.*

Source Code Reengineering Reference Model (SCORE/RM)

- Interpretation.
  - The process of deriving the meaning of a piece of software is started in this layer. The interpretation layer performs the following functions:
    - **Functionalization:**
      - *This is additional rationalization of the data and control structure of the code, which (i) eliminates global variables and/or (ii) introduces recursion and polymorphic functions.*
    - **Program reading:**
      - *This means annotating the source code with logical comments.*

Source Code Reengineering Reference Model (SCORE/RM)

- Abstraction.
  - The annotated and rationalized source code is examined by means of abstractions to identify the underlying object hierarchies: The abstraction layer performs the following functions:
    - **Object Identification:**
      - *The main idea in object identification is (i) separate the data operators and (ii) group those data operators with the data they manipulate.*
    - **Object interpretation:**
      - *Application domain meanings are mapped to the objects identified above.*

Source Code Reengineering Reference Model (SCORE/RM)

- Causation.
  - This layer performs the following functions:
    - **Specification of actions:**
      - *This refers to the services provided to the user.*
    - **Specification of constraints:**
      - *This refers to the limitations within which the software correctly operates.*
    - **Modification of specification:**
      - *The specification is extended and/or reduced to accurately reflect the user's requirements.*

Source Code Reengineering Reference Model (SCORE/RM)

- Regeneration.
  - Regeneration means re-implementing the source code using the requirements and the functional specification. This layer performs the following functions:
    - **Generation of design:**
      - *This refers to the production and documentation of the detailed design.*
    - **Generation of code:**
      - *This means generating new code by reusing portions of the original code and using standard libraries.*
    - **Test generation:**
      - *New tests are generated to perform unit and integration tests on the source code developed and reused.*

Source Code Reengineering Reference Model (SCORE/RM)

- Certification.
  - The newly generated software is analyzed to establish that it is operating correctly, performing the specified requirements, and consistent with the original code:
    - **Validation and Verification:**
      - *The new system is tested to show its correctness.*
    - **Conformance:**
      - *Tests are performed to show that the renovated source code performs at the minimum all those functionalities that were performed by the original source code.*


Thank You!