Chapter 5

Process Model for Round-trip Engineering with Relational Database

Leszek A. Maciaszek
Macquarie University, Australia

Iterative and incremental development of client/server database systems requires a round-trip engineering support, in particular in a design-implementation cycle. This paper identifies some more difficult round-trip engineering scenarios and defines processes needed to handle those scenarios. The processes conform to the current state-of-the-practice in forward and reverse engineering with relational databases.

The paper identifies limitations of a tool-driven round-trip engineering. The limitations can be linked to three reasons: (1) the inability of a CASE/4GL tool to always generate correct incremental code after schema has been changed, (2) the need for a CASE/4GL to understand the reverse-engineered procedural parts written (or modified) in the implementation phase, (3) the requirement that a database content (extension) be re-instated at the end of each design-implementation cycle.

Technical limitations introduce a risk that design models and a database implementation become misaligned and the design-implementation cycle cannot be continued for iterative and incremental software production. Project managers need a process model to impose necessary rigour on design and programming teams to alleviate technical restrictions. The paper defines a project management strategy that enforces appropriate automated and manual processes on database development teams.


This chapter appears in the book, Successful Software Reengineering by Sal Valenti. Copyright © 2002, IRM Press, an imprint of Idea Group Inc.
INTRODUCTION

Modern software development processes are invariably incremental and iterative. System models are refined and transformed through analysis, design and implementation phases – details are added in successive iterations, changes and improvements are introduced as needed, and incremental releases of software modules maintain user satisfaction and provide important feedback to modules still under development. As Rational Unified Process states: “An iterative process is one that involves managing a stream of executable releases. An incremental process is one that involves the continuous integration of the system's architecture to produce these releases, with each new release embodying incremental improvements over the other” (Booch et al., 1999, p.33).

Iterative and incremental processes need a strong round-trip engineering support between adjacent development phases. This is particularly true for lower engineering processes – design and programming phases. Changes in design models have to be forward-engineered to existing implementation and changes in implementation have to be reverse-engineered to design models.

In this paper, we determine the limitations of commercial automation to support round-trip engineering between a database design model and an incrementally implemented relational database. We identify various incremental changes to design and implementation, and we show how they can be round-trip-engineered. The changes include declarative and procedural aspects of database intention (schema). We require that round-trip engineering is constrained by the database extension, i.e., the latest database content must be re-instated in a new database. We define processes that have to be imposed on the design and programming teams so that round-trip-engineering can be properly managed. The process management aspect can be enhanced if a change monitoring system is implemented in the database and if it is itself a subject of round-trip engineering (so that a record of design and implementation changes, still subject to round-trip engineering, is kept current at all times).

BACKGROUND AND RELATED WORK

The objective of round-trip engineering is to support evolutionary development of software systems. The term was coined, I think, by Grady Booch who defines it as combining of forward code generation and reverse engineering that gives "...the ability to work in either a graphical or textual view, while tools keep the two views consistent." (Booch et al., 1999, p.16).

Round-trip engineering is concerned with an evolutionary development of new systems and it therefore differs from re-engineering which examines and alters a legacy system to recover its design and re-implement it in a new form. Neverthe-