Graph-based Reengineering

Based on research in Leg2Net and Sensoria, with contributions from Carlos Matos (ATX) and Fawad Qayum

Industrial Advisory Board, 25 September 2009
Does your SW look old and tired?

Change in business requirements and technology drive evolution of software.

- Business driven
  - Mergers and acquisition of companies
  - New legislation
  - Optimisation of business process

- Technology driven
  - Adoption of OO
  - Support for web access
  - Migration to SOA

- Life-cycle driven
  - Maintenance
  - Software aging

→ Time for the anti-aging treatment ...
Why is this difficult?

- Variety of platforms and architectures;
- Mappings between them such as
  - Cobol → Java
  - Oracle Forms → .NET
  - VB6 → VB.NET
  may require complex restructuring
- Need to preserve (some aspects of) application semantics
- Each project is different; legacy systems are large-scale and heterogeneous
Need a treatment that

- Can be applied repeatedly in different circumstances, with consistent results
- Provides automation (to allow scalability)
- Have reusable ingredients (to support different platforms, tasks)
Outline

- Motivation
- Main principles
- Application to SOA Migration
The Classical Rejuvination Approach: Horseshoe Model of Reengineering
Main Principles of our Approach

- Graph-based abstraction for scalable representation
- Graph transformation for high-level evolution planning
- Relations for keeping track of mapping with source code

Reverse Engineering
Redesign Transformation
Forward Engineering

Source System
Source Model
Target Model
Target System

R1
R2
R3
Graph Transformation for High-level Evolution Planning

Model redesign operations as graph transformations
- Semantic annotations to control rule application
- Search design space for best solution

Diagram showing graph transformation rules with semantic annotations.
Architectural Migration

1. Code Annotation
2. Reverse Engineering
3. Redesign
4. Forward Engineering

- Generic, has to be instantiated (and iterated)
Outline

- Motivation
- Main principles
- Application to SOA Migration
Reengineering towards Services: Two Dimensions

- Functional
  - UI
  - Logic
  - Data

- Technical
  - F1
  - F2
Migration to Layers

Client-Server to 3-Tier

Client

Server

UI

BL

Data
Migration to Services

3-Tier to SOA
Two Instances

1. Client-server $\rightarrow$ 3-tier
2. 3-tier $\rightarrow$ SOA
   - Identifying services
   - Grouping of operations by interfaces (port types), services, components
ATX Care Studio: Code and Query Pattern
Conclusion

- Technology-independent approach to model-driven reengineering: Map model-level to code-level transformations
- Graph transformation as technology for model-level transformation
- Scalable due to automation and abstract graph model
- Iteration allows to decompose complex transformations into ones
Future Projects?

- Working prototype as proof-of-concept based on Java

- Commercial exploitation would require
  - support for legacy languages
  - more tool development and integration

- ATX could be ideal industry partner, but project is scientifically incremental, commercially risky