PHILIPS

Bootstrapping Domain-Specific Model-Driven Software Development within Philips

Marc Stroucken, Hans Jonkers, Richard Vdovjak
Philips Research
Contents

• Background
• VAMPIRE Approach
• Requirements for MDD
Background

- Research started in 2002
- First working environment in 2003 based on:
  - Microsoft Visual Studio .NET 2002 – C#
  - Altova XML Spy – Authentic – XML
- Developed incrementally: current version 6
- Applications
  - Interface Specification (CBA, ISpec)
  - Documentation generation
  - Strongly-typed C# code generation (performance, memory-footprint)
  - Graphical application design
VAMPIRE Approach

Visual Agile Model-driven Product-development an Integrated Research Environment

Basic Design Pattern
(ObjectModel = DSM)
VAMPIRE Approach

ObjectModel as instance of a MetaObjectModel (metametamodel)

Editor -> Model data -> Generator

Meta ObjectModel

Object Model

---

input/output
instance of
uses
VAMPIRE Approach

ObjectModel Creation

input/output
instance of
uses
VAMPIRE Approach

- Meta ObjectModel
- Object Model
- C# Code Generator
- Editor
- Model
- Generator

Code Generation

- input/output
- instance of
- uses
VAMPIRE Approach

Philips Research

Bootstrapping DSM within Philips – Marc Stroucken, Hans Jonkers, Richard Vdovjak
VAMPIRE Approach

Editor/Generator Implementation
VAMPIRE Approach

MetaObjectModel DLL

input/output
instance of
uses
VAMPIRE Approach

Bootstrapping problem
Code generator both creates and uses the MetaObject-Model DLL
VAMPIRE Approach

Editor Generation
(comparable to DSL)
VAMPIRE Approach

ObjectModel as Data
VAMPIRE Approach

Interpretive WoW
(New Editor/Generator Implementations using Model Reflection)
VAMPIRE Approach

Aspect-Oriented Modeling (AOM)
(Extending MetaObjectModel towards a Domain-Specific Metametamodel)
Requirements for MDD

- Fast development of models, editors, generators (days)
- Easy programming model (e.g. no intermediate formats)
- Customizable code generation (performance, memory)
  - XML configurability (legacy), not limited to one root
  - Loosely coupled tools
  - Easy generator composition (networks)
  - Metamodels as objects
  - Model reflection vs. programming language reflection
  - Metametamodel extension (bootstrapping)
  - Aspect-oriented modeling (AOM) and model composition
  - Interpretive way-of-working vs. code generation
  - “Unknown” attributes
  - Multiple inheritance