Modeling that works with code

November 12, 2009 – InfoSupport Mechelen

Marcel De Vries

Upcoming sessions

- ~~October 8 (Crowne Plaza Antwerp)~~

  ~~*C# 4.0 and Testability (Dino Esposito)*~~
- ~~November 3 (Microsoft Belgium)~~

  ~~Stubs – Lightweight Test Stubs and Detours for .NET~~
- **November 12 (InfoSupport Mechelen)**

  Modeling in VSTS2010 (Marcel De Vries)
- November 26 (KDGH Antwerp)

  .NET RIA Services
- December 17

  F#, Functional programming for .NET (Kurt Schelfthout)

Extra BIG event

December 4, 2009

Organization by Microsoft & VISUG

Scott Guthrie

More details soon at www.visug.be

usg innotiv professionals in ict

ctg

EURICOM .NET architects
A DIMENSION DATA COMPANY

avanade

ORDINA
CONSULTING | ICT | OUTSOURCING

sdworx

AUSY

Microsoft

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Sparkles
IT Consultancy & Training

neomatics

JetBRAINS

digipoint
passion inside

architects for business & ict
ae

TEAMLINK

ps_testware
Your devil's advocate

REALDOLMEN

telerik
deliver more than expected

Modeling that works with code

Marcel de Vries

Technology Manager Microsoft

marcelv@infosupport.com

Agenda

- Why we use models
- Models first approach
- Code first approach
- Conclusion

Why we use models & diagrams

Communication

Why software models are challenging

- Creating models & diagrams costs significant effort
- They should be used with common sense
- UML as a standard in the market
  - Only few models are used
  - Usage varies by person
    - E.g. Use case modeling can be done in many different ways
  - You need to add your company standard to the way models and diagrams are created

Why do models become stale?

- Models are often stale
  - What you design today will be implemented differently
- Creating and maintaining a model takes time and effort
  - You create them because they bring value
  - But how can you keep enough value in a model so you start maintaining it?
  - Create additional value
    - Use it for generation
    - Use it for validation
- Ultimate goal: Run a model
    - Oslo vision

When do you create a model?

- Depends on many different factors
  - Available tools
  - E.g. Number of team members
- It is about communication, so:
  - Create only models that serve a purpose
    - If your customer don't (want to) understand Use Cases then use a different technique!
  - Create only models of those things that differ from the normal case
    - Create a diagram only once to explain a pattern
    - Don't model until you drop
    - If generation is the goal, think real hard how to make it work
- Two approaches to use models
  - Model first
  - Code First

Models first approach

- Scenario: You start a new project and start defining the solution you're going to build
  - **Requirements Analysts** use Activity and Use case diagrams to outline functionality
  - **Architects** build logical diagrams to outline the required architecture to meet customer demand stated in the requirements
  - **Testers** use the diagrams to do it's test analysis based on the requirements specified
  - **Developers** can use the diagrams to materialize the projects
- Models are used to communicate throughout the team what we are going to build

Visual Studio 2010 Architect edition

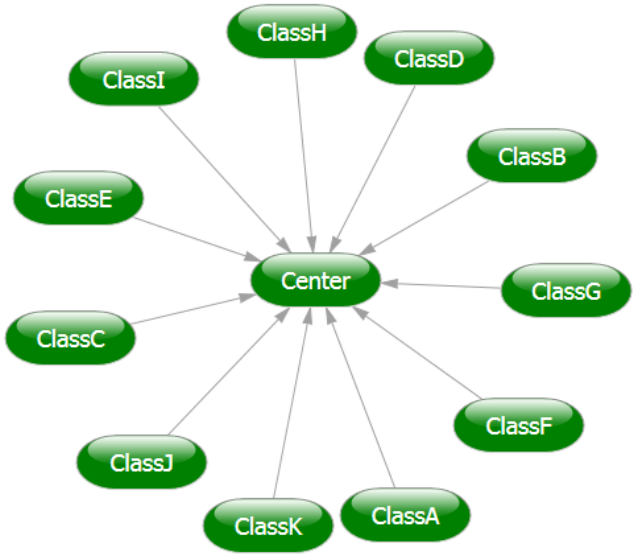
- New project type: Modeling Project
- Support for UML 2.x Diagrams
  - Use case
  - Sequence
  - Component
  - Activity
  - Class
  - UML Profiles
- Other new type of diagrams
  - Layered Diagram
  - DGML Diagrams in many different forms

Demo

Using Modeling projects in VS2010

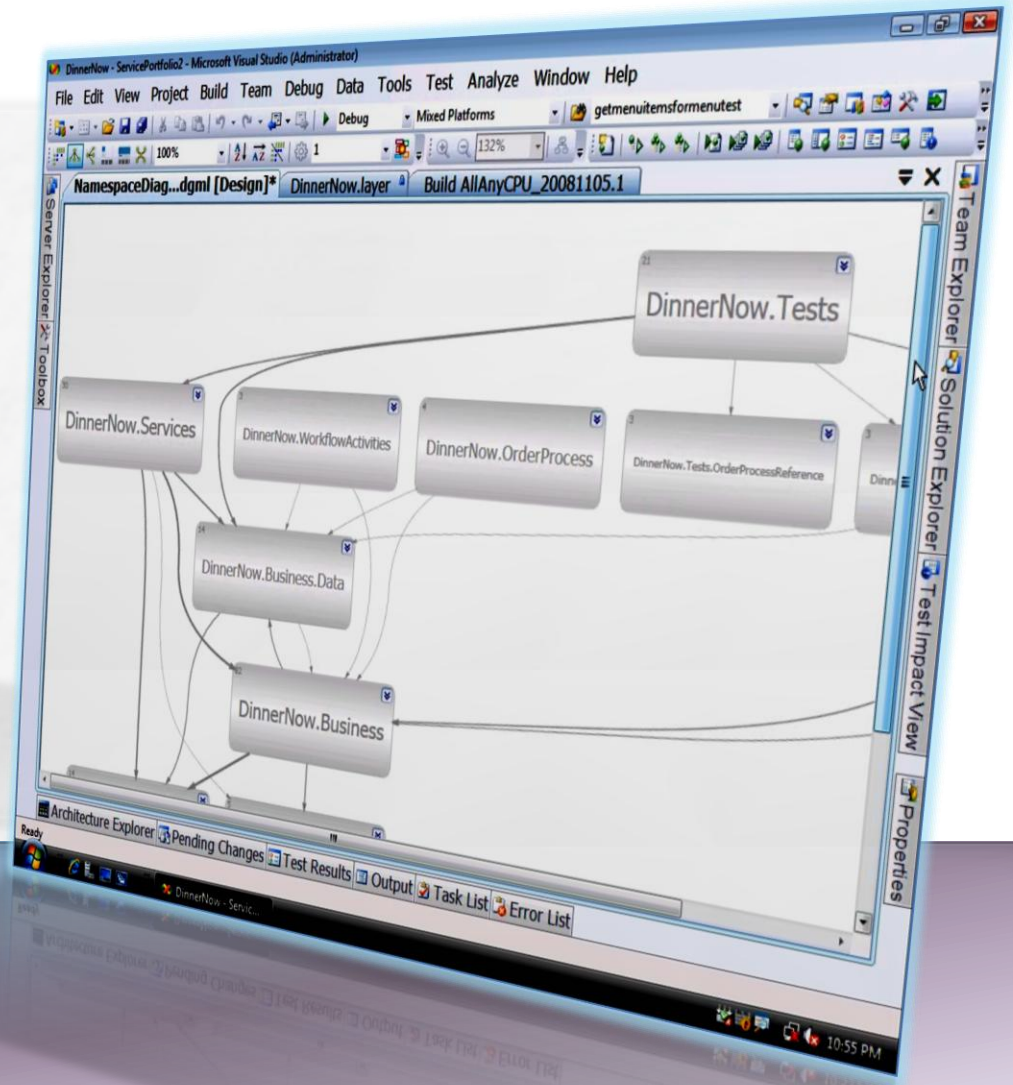
Code First approach

- Scenario: maintenance on existing projects, extending an existing codebase
- Developer can generate diagrams to gain insights on code dependencies
- Architect can find architecture patters
- Based on visualizations
  - New type of diagram: DGML
  - **D**irected **G**raph **M**arkup **L**anguage
- Create your UML diagrams on demand!



Architecture Explorer

- Understanding a system can prevent the butterfly effect.
- Architecture Explorer helps discover and understand how a system works.
- Visualize existing code assets and how they connect.

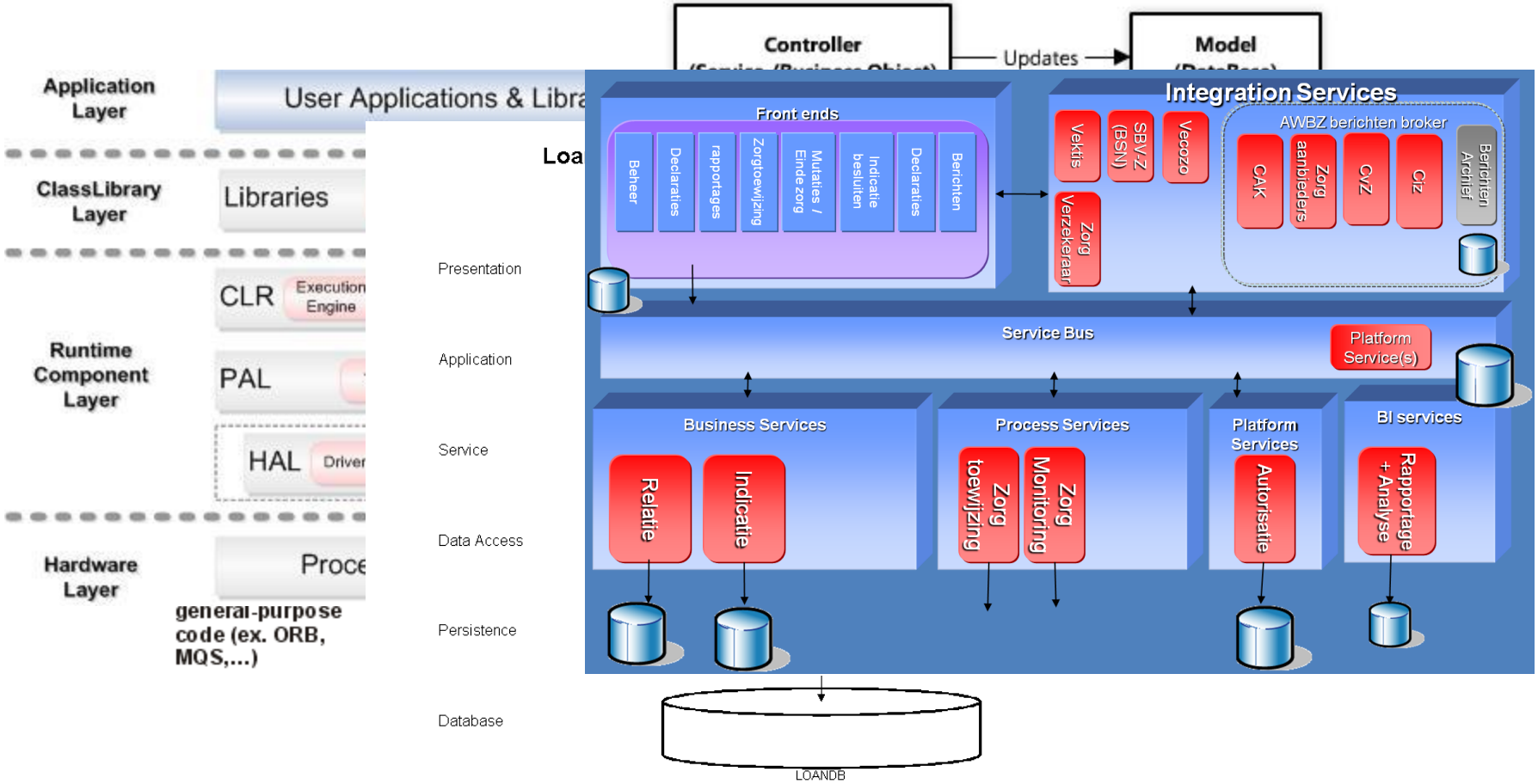


Demo

Code exploration

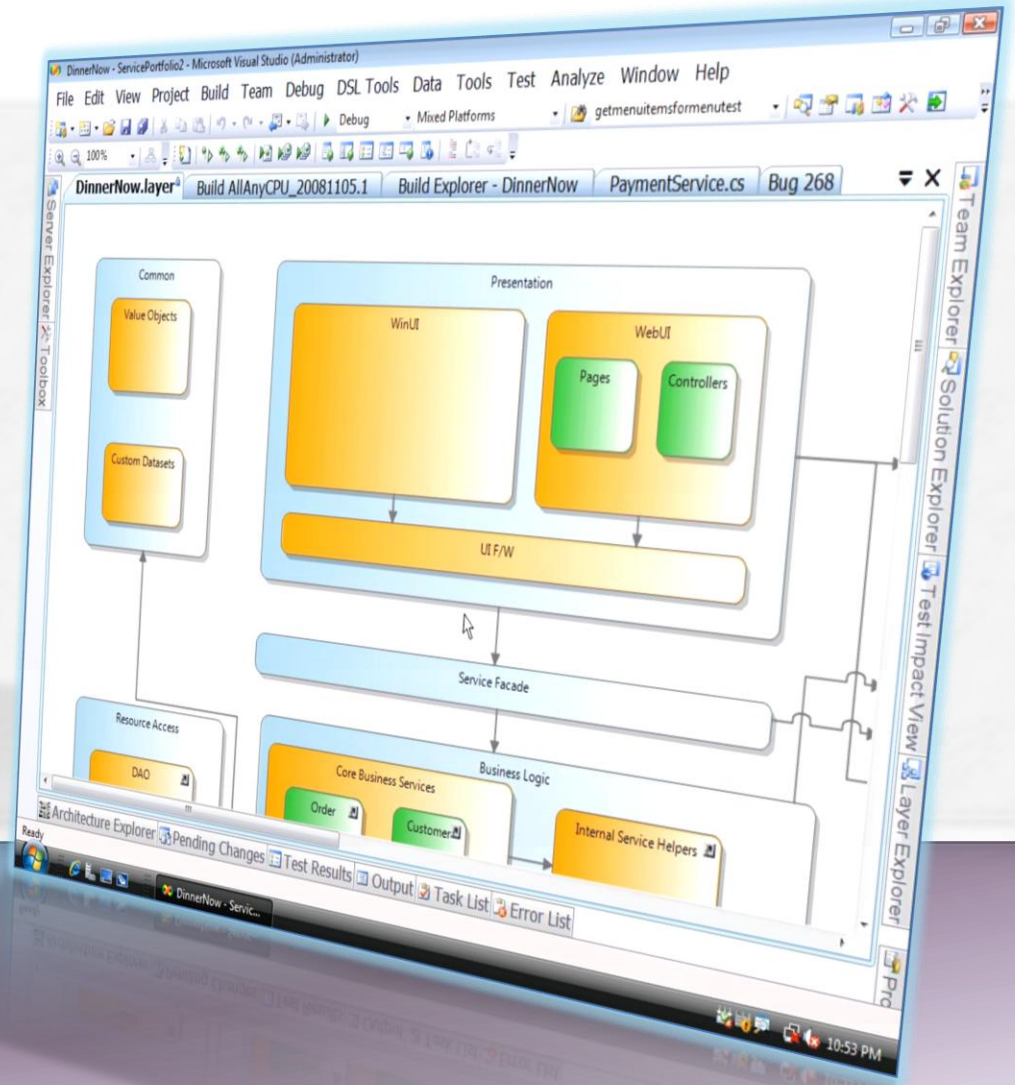
Generating Sequence Diagrams

Models that work with code



Layer Diagram

- Architectural validation ensures code matches intended design.
- Layer diagram details the intended design.
- Classes and namespaces are mapped to layers in the diagram.

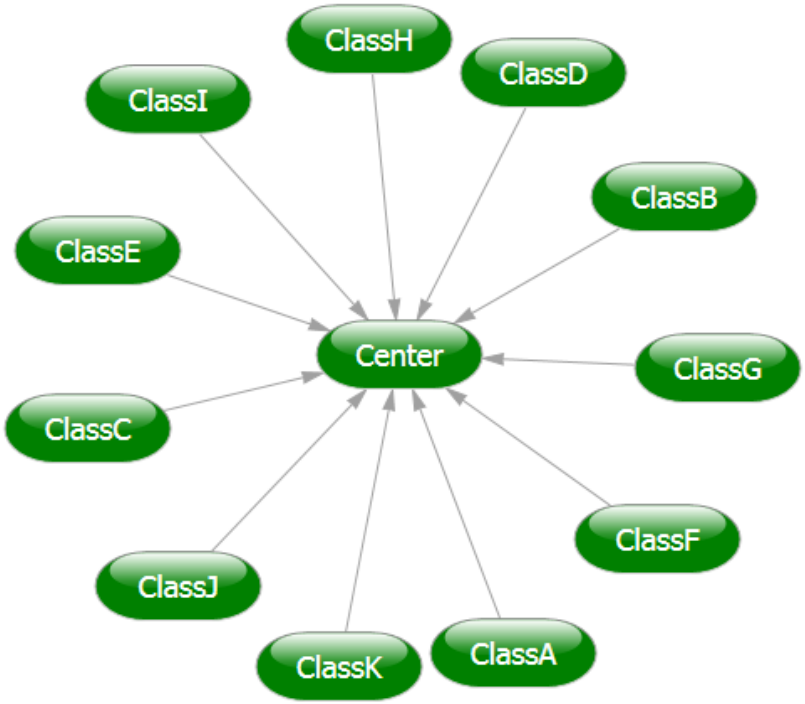


Demo

Layered diagram and validation

DGML

- Directed Graph Markup Language
- Nodes
- Containers
- Links
- Style



- You can generate your own Diagrams!
  - Use progression API

Demo

Creating Custom DGML Diagrams

To Answer some questions:

- How can you use models from the start of your project and let them reflect what you are actually building, instead of reflecting what you wanted to build?
  - **Add value to the models beyond pretty pictures**
- How do you keep track of software artifacts that come from certain requirements stated in e.g. Use case stories?
  - **Use Work item Links**
- How can I make sure that the product I designed is actually the product that is build and how am I able to verify this during the daily build cycle?
  - **Use Layered Diagram**
- On the other hand how can I understand the architecture of a product I need to maintain that has grow evolutionary and has no design available?
  - **Use architecture explorer**
  - **Use Reverse engineering options**
- How can I analyze the architecture of such a software product and get insights in the quality of that design in the codebase?
  - **Create custom DGML reports**

Conclusion

- Use models and Diagrams to communicate in your team
- VS2010 now supports both approaches to modeling
  - Models First
  - Code First
- New innovations in models that work with code
  - Layered diagram to validate your designs
  - DGML to create new visualizations of your code