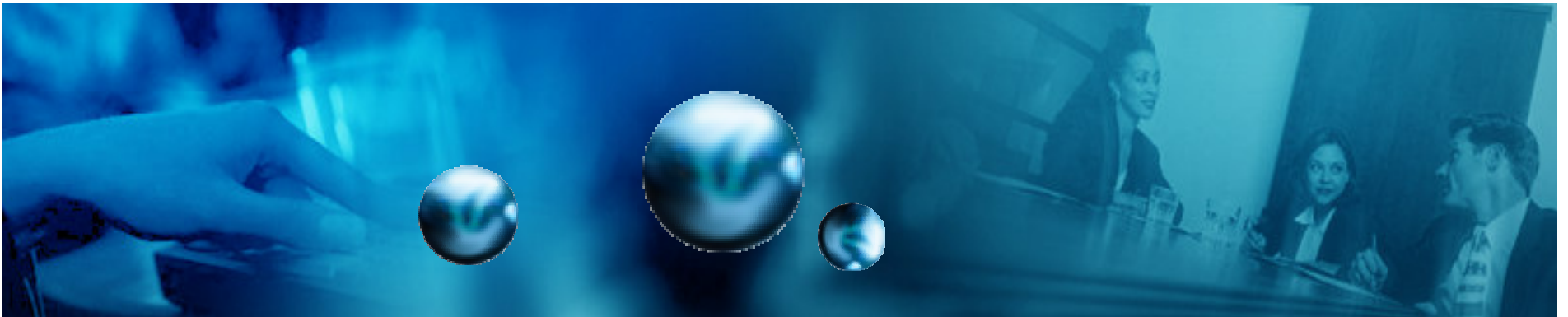


# What is Software Design?



The Model is the Design – A Pragmatic Approach

Ben van der Merwe, CT-SPIN, 17 August 2005





# A Model as the Design

What is a Software Design Model?

Dictionary definition for *Model*:

“A preliminary work or construction that serves as a plan from which a final product is to be made: *a clay model ready for casting.*”

For *Design*:

“A graphic representation, especially a detailed plan for construction or manufacture.”

My definition for a *Design Model*:

“**Preliminary representation** of an intended product that serves as a **plan** from which the final product results, through some **process.**”



# What is the Role of a Design?

That's really what you have to decide:

- *Describes what resulted?*
- *Describes what should result?*

Key Potential Role: Communication

There may be many stakeholders:

- Customer (GUI, Deployment, Architecture)
- Partners
- Teams
- Developers
- Management
- Documentation Team



# Why the Code is not the Design

- Code are *instructions*
  - Understood by the **compiler** and **translated**
  - The CPU translates and executes
- Detailed Instructions obviously not effective for human communication
- Inappropriate **level of detail** (stakeholders)
  - “Not seeing the Forest for the Trees”
    - C++ can be faster than hand-coded assembler
  - Flow? Structure? Execution State?
- **Hidden information**
  - Structure, Modules (jars, dll's, packages)
  - Configuration
- **Wrong View**
  - For Architecture, Interaction, Deployment , GUI



# Trends – It's not just code anymore

- Proliferation of Configurability
  - Example: .Net Remoting (contrast with Java RMI)
- Component-based programming, Re-use
- Code Generation
  - For Object-Persistence, from XML definition
- Domain Specific Languages
  - XSD for XML
  - SQL
  - Visual Studio 2005 DSL Infrastructure
- Rule Engines
  - Externalizing business logic and configuration
- Software as Services (SOA)
  - Based on published interfaces, defined by a schema



# Some Observations

- Developing Software today:
  - We engage in many kinds of activities
  - We are producing many different artifacts
  - We must communicate between many different stakeholders
  - Deployment and configuration becoming more complex
- Challenges:
  - *Understanding and communicating* what is required
  - *Deploying, performing and maintaining*
- How do we manage all of this?
- Where is the documentation / visualizations / designs to assist us?



# Models to the Rescue

- Domain Model
  - Develop an understanding of the business domain
  - Expert knowledge capturing
  - Capture constraints, business rules
  - Develop a common **language**
  - Entry into OO design
- Deployment Model
  - Start planning for deployment early!
  - Resource allocation, budgeting
  - High Availability planning
  - Impacts on performance
  - Impacts on *architecture and design*



# MDSD

## Model Driven Software Development

- Domain analysis and software product line engineering
- Meta modeling and domain-specific modeling languages
- Model driven code generation
- Template languages
- Domain-driven framework design
- The principles of agile software development



# For More Information

*Model-Driven Software Development*

[http://www.mdsd.info/mdsd\\_cm/headlines.php](http://www.mdsd.info/mdsd_cm/headlines.php)

*Agile Modeling (AM) Home Page: Effective Practices for Modeling and Documentation*

<http://www.agilemodeling.com/>

*Visual Studio 2005 Team System Modeling Strategy and FAQ*

<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnvs05/html/vstsmode.asp>