zeligsoft

Model-Driven *:
Beyond Code Generation

John Hogg
COO, Zeligsoft

2004-05-20

Version 1.0

Introduction

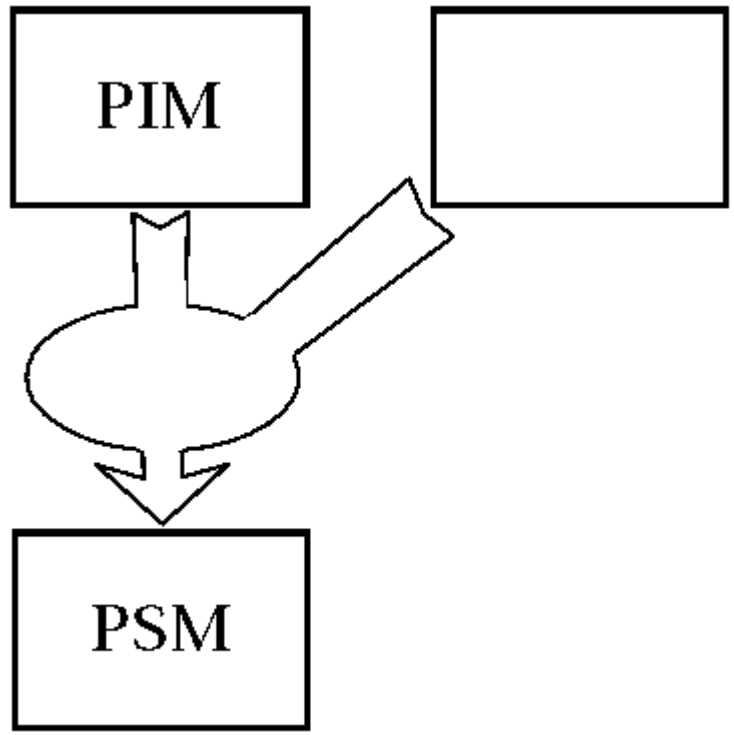
- MDA is more than a CIM, a PIM and a PSM
- MDA is a pattern
- MDA is a philosophy

  work at the model level wherever it makes sense to do so

- Philosophy
- Examples
- Success factors

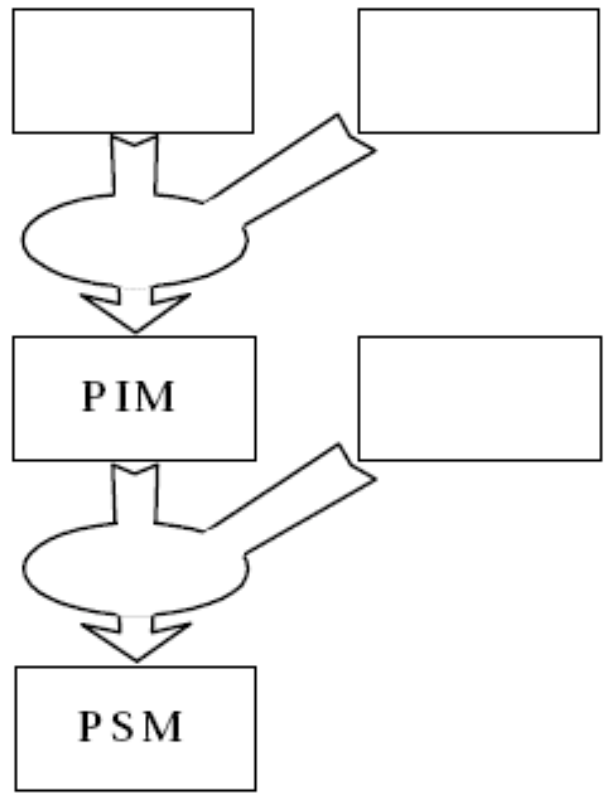
MDA Popular Misconceptions

- "MDA" is the process of applying a transformation to convert **the** PIM (Platform Independent Model) to **the** PSM (Platform Specific Model)
- There is one PIM
- There is one PSM
- PIMs and PSMs must be different models
- When a PSM has been generated, MDA is over
- *None of these statements are true!*



Selling the Wrong Thing

- A PIM is a PIM *with respect to* a class of platforms
- A PSM is a PSM *with respect to* a platform from that class
- A PSM may be a PIM with respect to another platform class
- *MDA is a pattern*
- But that's just the beginning…



Why MDA? (1 of 2)

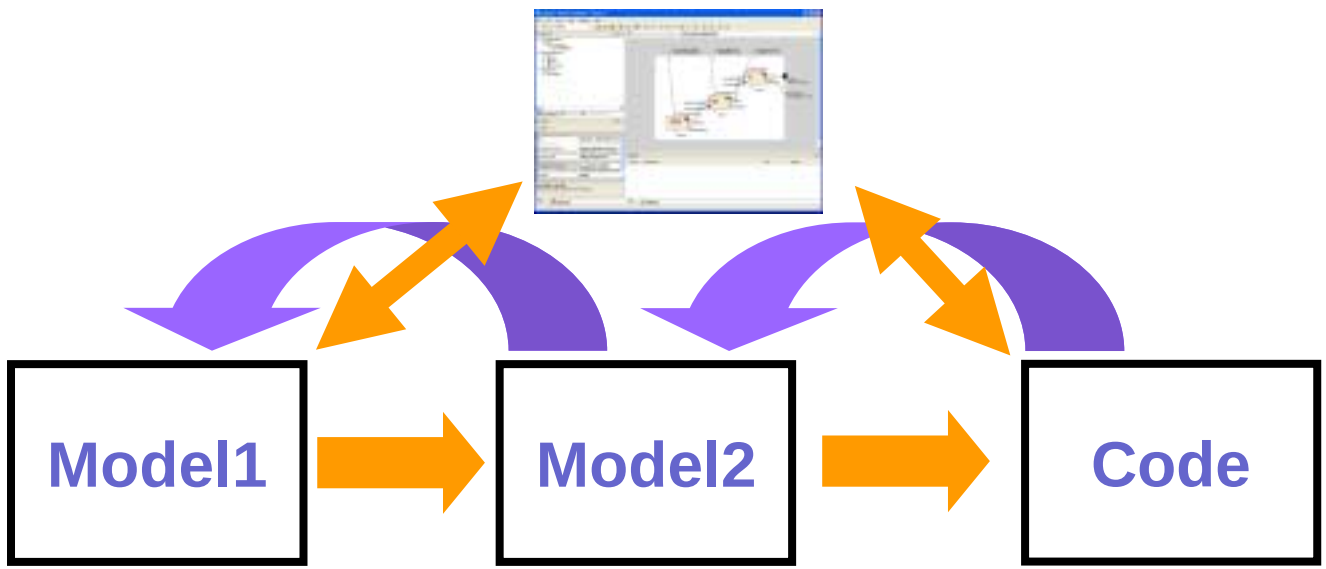
- To separate domain and platform concerns?
  - No—that's a mechanism, not a goal
- Abstraction-related reasons:
- To improve communication
  - Use notation natural to stakeholders
- To increase flexibility
  - Retarget to different technologies as required
  - Overall architecture is more adaptable

Why MDA? (2 of 2)

- Automation-related reasons:
- To deliver systems faster
  - Through automated generation from powerful abstractions
  - Through reduced defect rate
- To increase quality
  - Details “correct by construction”
  - Developers can focus on important issues

Transformation Enables Connection

- Model transformation/code generation is an enabler
- Automated transformations enable precise mappings
- Precise mappings enable connections to tools and capabilities
- *This is orthogonal to platform independence*

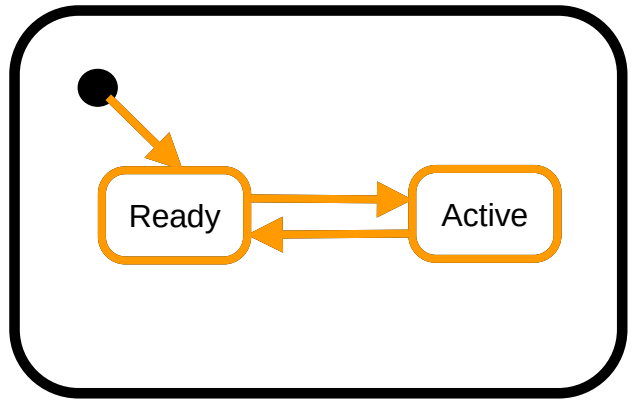


Model-Driven Opportunities

- Transformation enables tools
- Common properties:
  - Model-driven processing
  - Output usually presented in model form
- Benefits seen earlier:
  - Better communication/understanding
  - Better applications faster

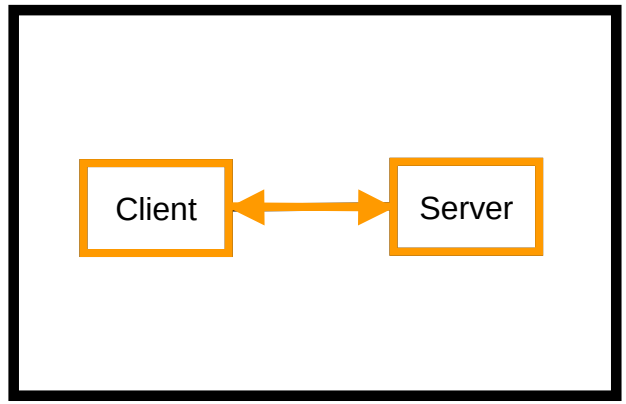
Model-Driven Debugging

- Generate debug information with code
- View and control execution at the model level
  - State machine execution



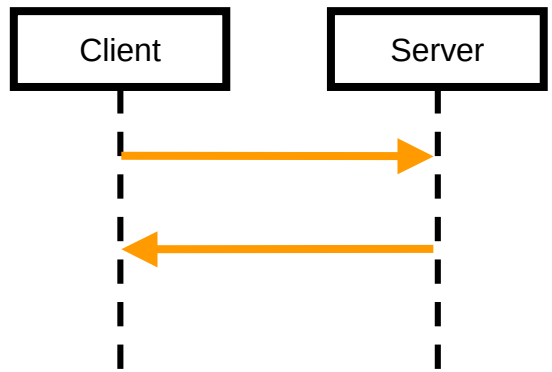
Model-Driven Debugging

- Generate debug information with code
- View and control execution at the model level
  - State machine execution
  - Object communication



Model-Driven Debugging

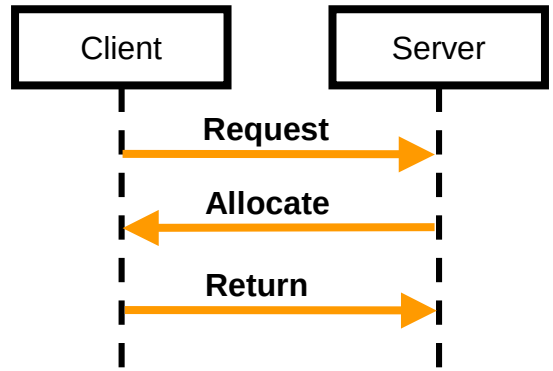
- Generate debug information with code
- View and control execution at the model level
  - State machine execution
  - Object communication
  - Sequence diagram
  - Other possibilities: activity diagrams,…
- Not “simulation”
  - Do you “simulate” a debug C++ execution?
- Not even “executable UML”
  - Do you write “executable Java”?

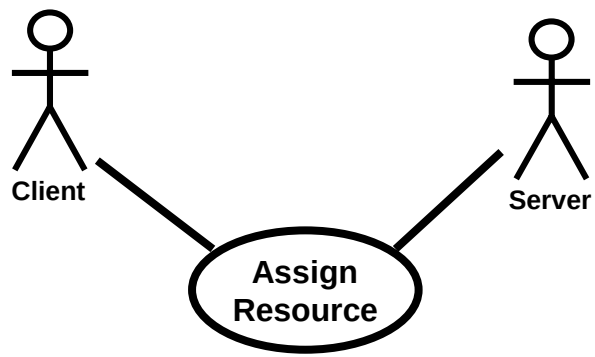


zeligsoft

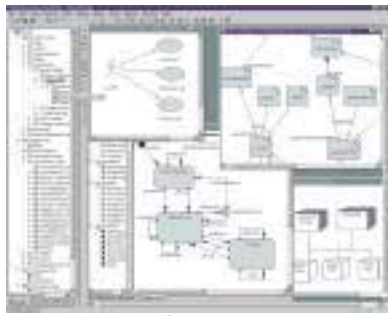
Requirements Modelling

- Model-driven system specification
  - Use case diagrams (models)
  - Sequence diagrams
  - Other possibilities:
    - Activity diagrams
    - Timing diagrams
- Requirements are communication tools in themselves
- They are part of MDA when they are used for Model-Driven Testing



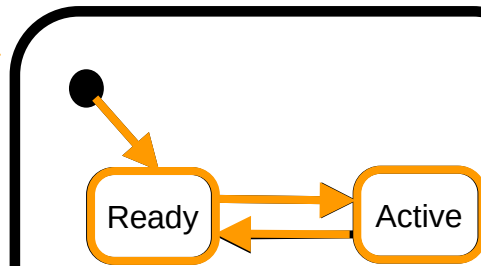


Model-Driven Testing

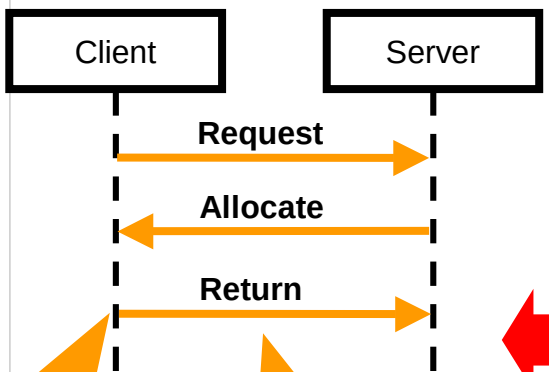


Application

Test generator

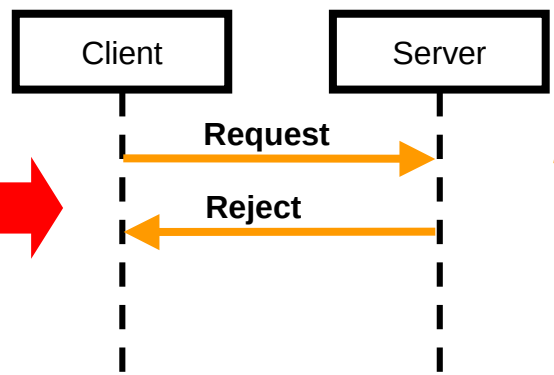


Execution



Test specification

Test driver



Model-Driven Static Analysis

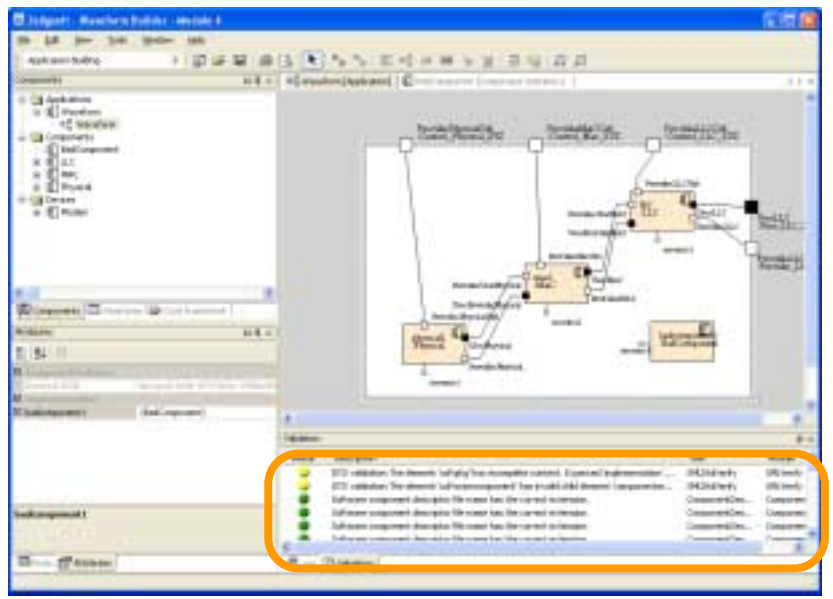
- "Static analysis": process/transform a model to understand properties
- Many examples:
  - Metrics
  - Formal verification
  - Validation
  - Schedulability
  - …
- We'll look at three

Model-Driven Analysis: Formal Verification

- Formal verification: mathematical analysis of models to statically prove dynamic properties
- Examples:
  - Deadlock, livelock
  - Reachability
  - Simultaneous state activation
  - …
- Shows great future promise
- Has shown great future promise for at least fourteen years
  - May show great future promise for the next fourteen years

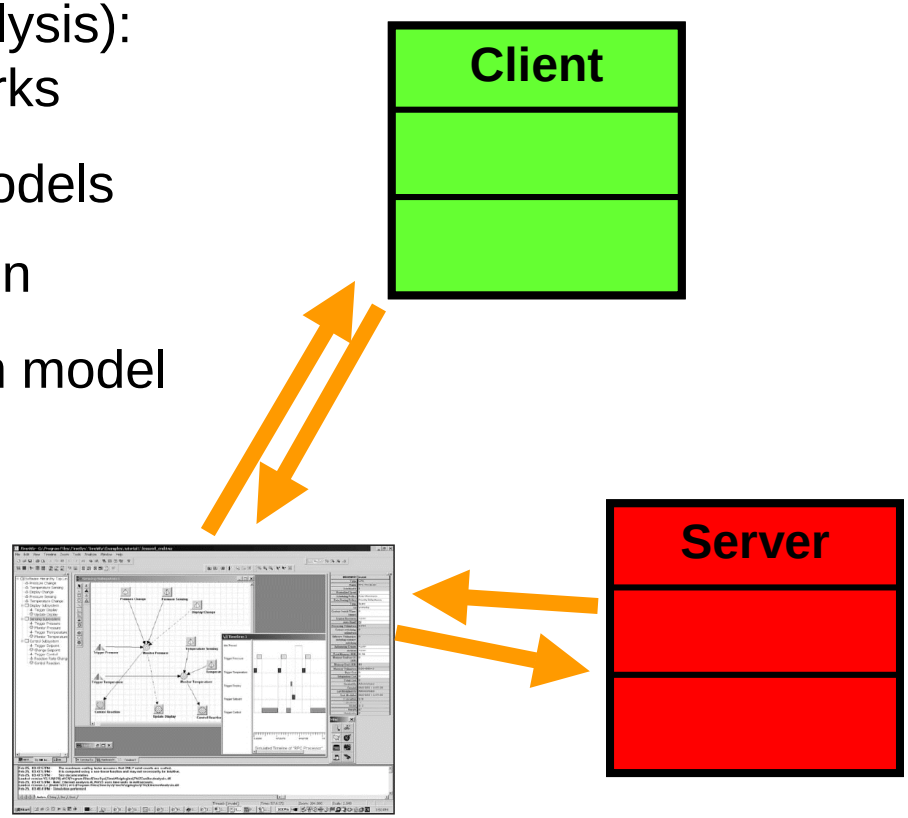
Model-Driven Analysis: Validation

- Validation: analysis of models to prove static properties
  - Much easier than formal verification!
- Examples:
  - Exactly one AssemblyController per Application
  - All mandatory properties assigned
  - Port types match
  - …
- Easy to do today
- “Low fruit”, high value



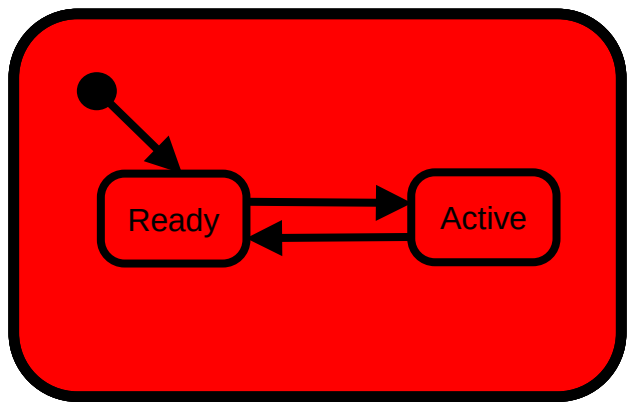
Model-Driven Schedulability Analysis

- Schedulability analysis: will a system meet its deadlines?
- RMA (rate-monotonic analysis): formal verification that works
- Can be applied without models
  - Models improve intuition
- Information extracted from model into analysis tool
- Information written from tool into model
- 2x2 MDA/RMA tool integrations today



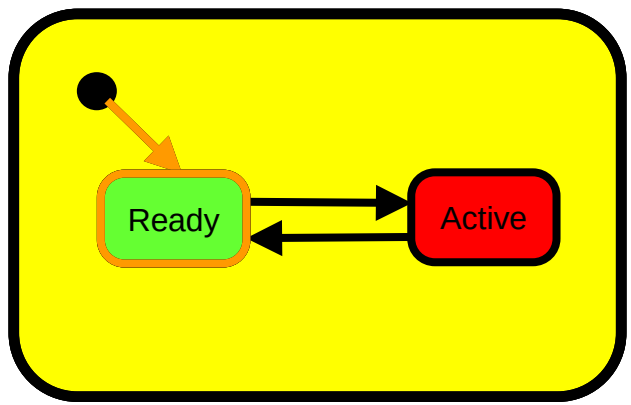
Model-Driven Run-Time Analysis

- “Run-time analysis”: collect model information at run time
- “Model-driven run-time analysis”: collect model-related information and display at the model level
- Example: model-driven coverage
- Can be shown at
  - Code level
  - State machine level (as here)
  - Class level
  - Structure level



Model-Driven Run-Time Analysis

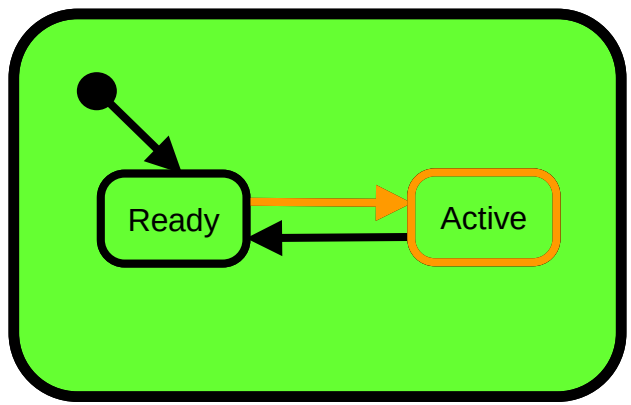
- “Run-time analysis”: collect model information at run time
- “Model-driven run-time analysis”: collect model-related information and display at the model level
- Example: model-driven coverage
- Can be shown at
  - Code level
  - State machine level (as here)
  - Class level
  - Structure level



zeligsoft

Model-Driven Run-Time Analysis

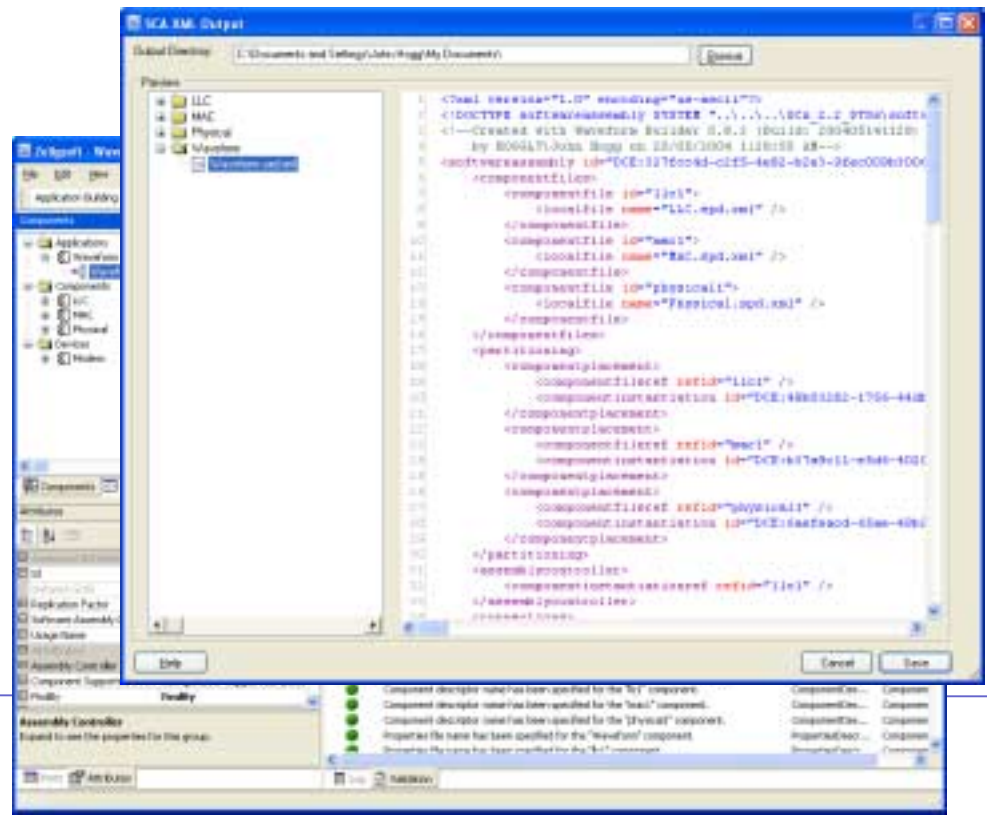
- "Run-time analysis": collect model information at run time
- "Model-driven run-time analysis": collect model-related information and display at the model level
- Example: model-driven coverage
- Can be shown at
  - Code level
  - State machine level (as here)
  - Class level
  - Structure level



zeligsoft

Model-Driven Configuration and Deployment

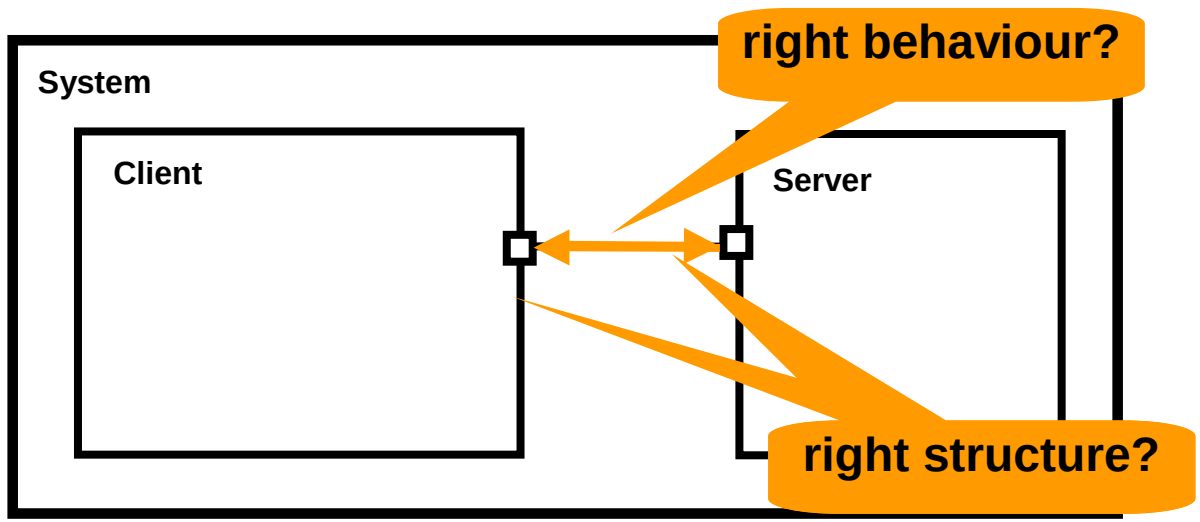
- Configuration descriptors: XML component specifications
- Not as glamorous as code
- Essential part of application delivery in some domains
- Build by hand…
- Or MDA
- Classic MDA with different “code” target



zeligsoft

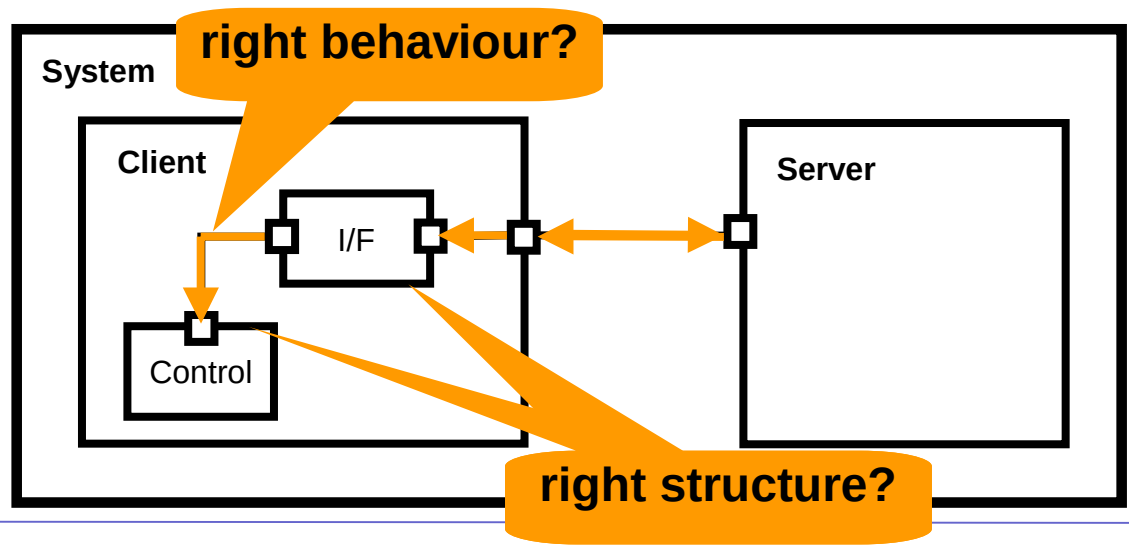
Providing the Right Value: MDA is Fractal

- MDA enables users to make *more* mistakes
  - Make mistakes quickly and cheaply
  - Explore a broader solution space
  - Select the best solution and move on
- Fast, cheap mistakes require partial models



Providing the Right Value: MDA is Fractal

- MDA enables users to make *more* mistakes
  - Make mistakes quickly and cheaply
  - Explore a broader solution space
  - Select the best solution and move on
- Fast, cheap mistakes require partial models



MDA is Fractal: Consequences

- Partial models=>information is missing
- Useful tools must provide results early in the process
- How to fill in the missing information?
- Add by hand
  - Performance estimates
  - Debugging behaviour
- Automatically generate from other parts of the model
  - Test frameworks
- Make those mistakes early!

Summary

- MDA is more than a CIM, a PIM and a PSM
- MDA is a pattern
- MDA is a philosophy

  work at the model level wherever it makes sense to do so

- Philosophy
- Examples
- Success factors

Thank You.

John Hogg
hogg@zeligsoft.com