Karl Frank

Principal Architect: Product Strategy and Architecture
kfrank@borland.com

# OMG™ Workshop

# MDA

## Tool Chains for MDA™?

Let's consider leaving our tool chains behind us.

Please note the existence of and make use of the **speaker notes**.

Borland®

# Agenda:

Origins of MDA™ in relationship to development tools

The fundamental insight behind MDA implies a need for visibility into and across projects.

Toolchains block this visibility!

This is the substance of the talk, next items covered briefly:

- Early sightings:  MDA-like features before "MDA" meant MDA
- MDA in relationship to Microsoft's vision of Software Factories
- First round of products (Actually mentioning competitor work)
- A view of the future: Roles, Projects, and the Logical Repository





kfrank@borland.com

# Assumptions:

We all understand that

MDA is not the same as MDD

- Objectory, OMT, Fusion, RUP, ROOM, the list goes on and on
- All are model-driven development processes in some respects

MDA not a methodology, but an "approach"

No MDA standard, but many MDA standards

That among these, UML® is most often thought essential

- Yet many experts would put MOF, not UML® at the head
- And many others would put QVT ahead of both
  - Announcement: A 2nd revised submission on QVT on 03-02-05



kfrank@borland.com

# Origins of MDA and Development Tools

Objectives: Based on Insight into the Roots of MDA

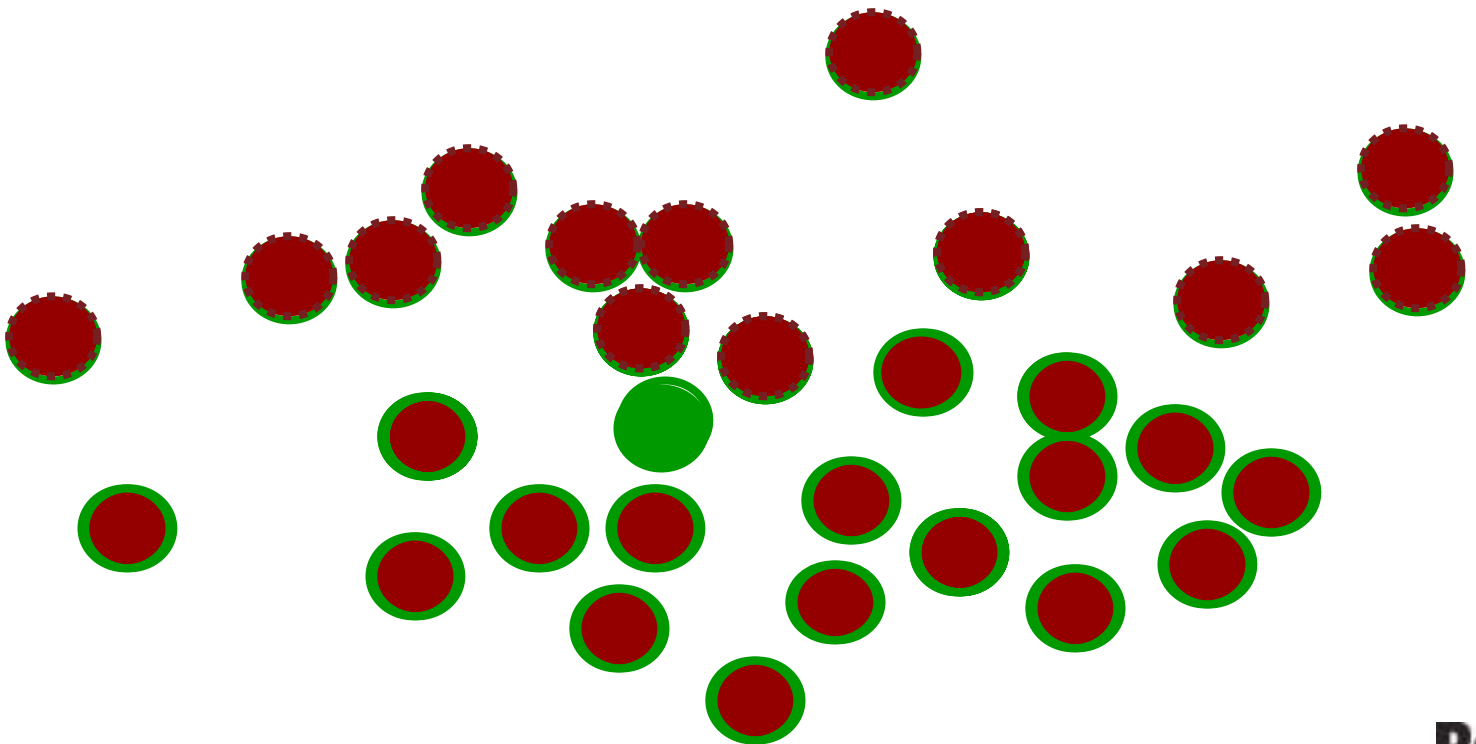
- To discover what MDA practice will imply for Tool Chains
- To understand MDA in relation to Software Factories

Borland®

kfrank@borland.com

# We could think of development as if..

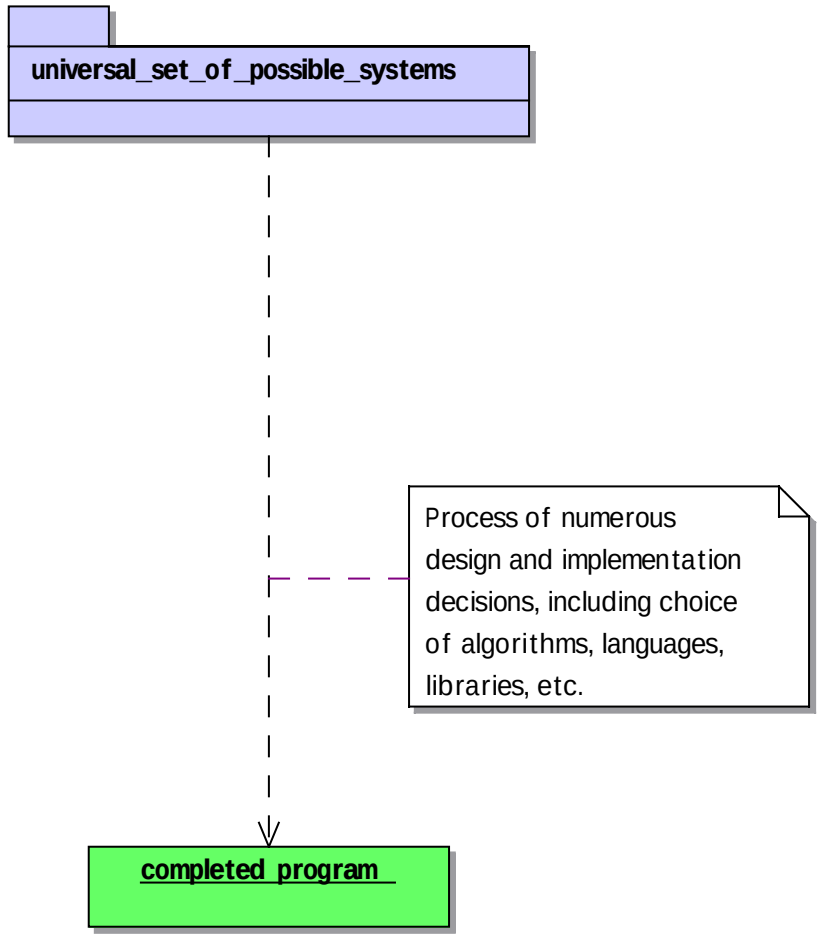
it were a stepwise process in which a conceptual universe of every possible program is narrowed by successive decisions on scope, design, algorithm, technology, etc. till one remains.



Borland®

kfrank@borland.com

# Diagrammatic view



# David Parnas in 1976

**Context:** Consultant to Software Cost Reduction Project

**Problem:** How best create new program similar to existing one?

**Observation:** *Usual approach to crafting a new program in the same domain as an existing program is to modify the source code. E.g. porting to new OS.*

**Insight:** *The development process (in an ideal case) takes the program through intermediate stages. In practice, these are abstract. At some stage the design is not dependent on OS..*

**Solution:** *Start again from that intermediate stage!*

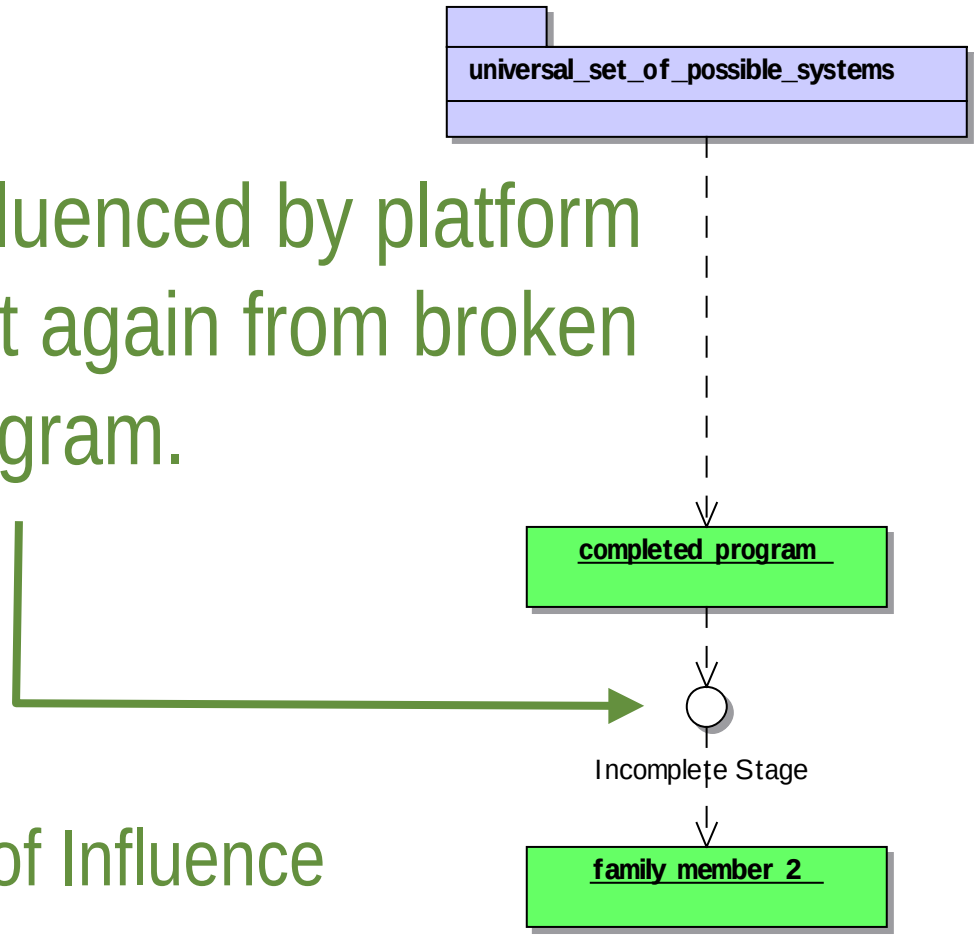
**Requires:** 1. *The right intermediate stage be preserved as a usable spec and* 2. *you must be able to find it!*

Borland®

kfrank@borland.com

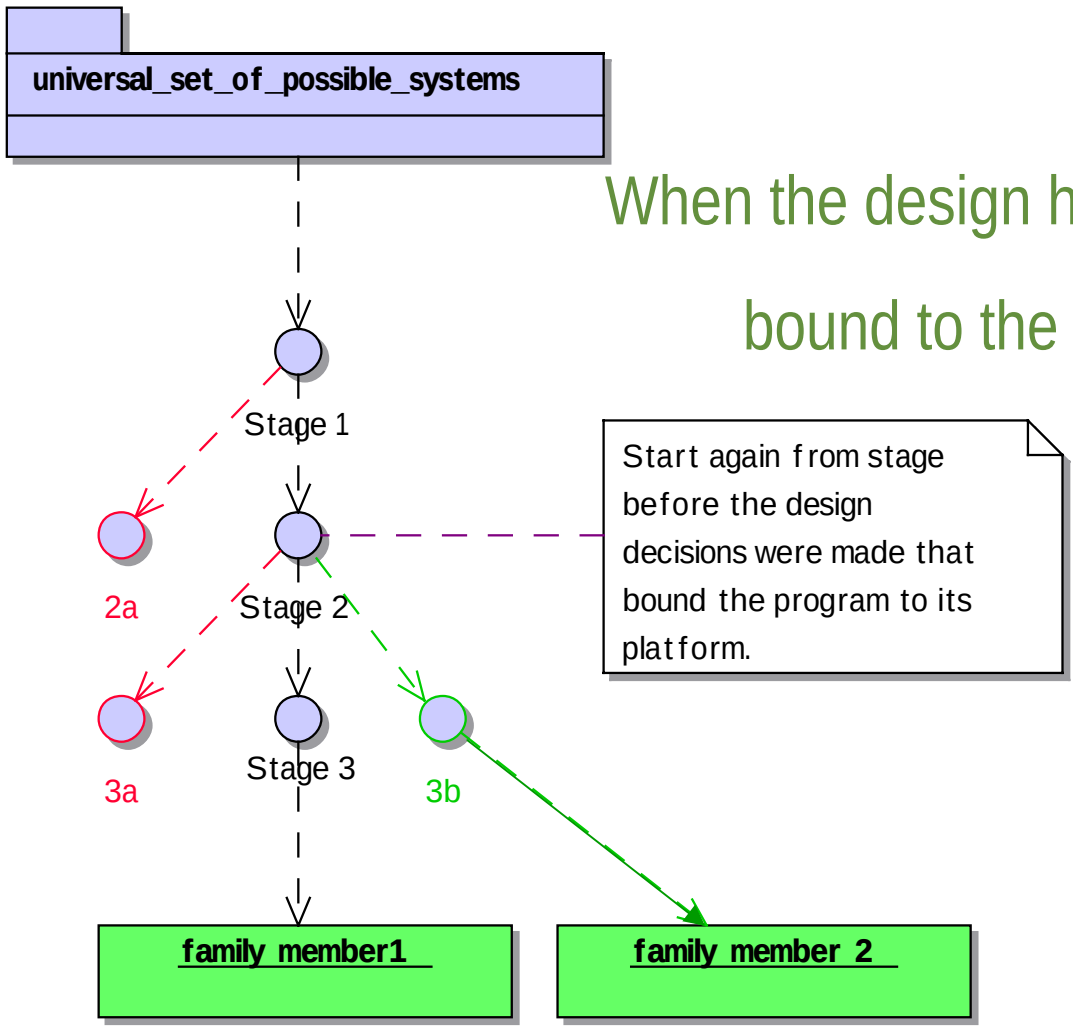
# The Usual Approach, still around today

erase all code influenced by platform decision and start again from broken program.




Problem: Locality of Influence

# There must be some intermediate stage...

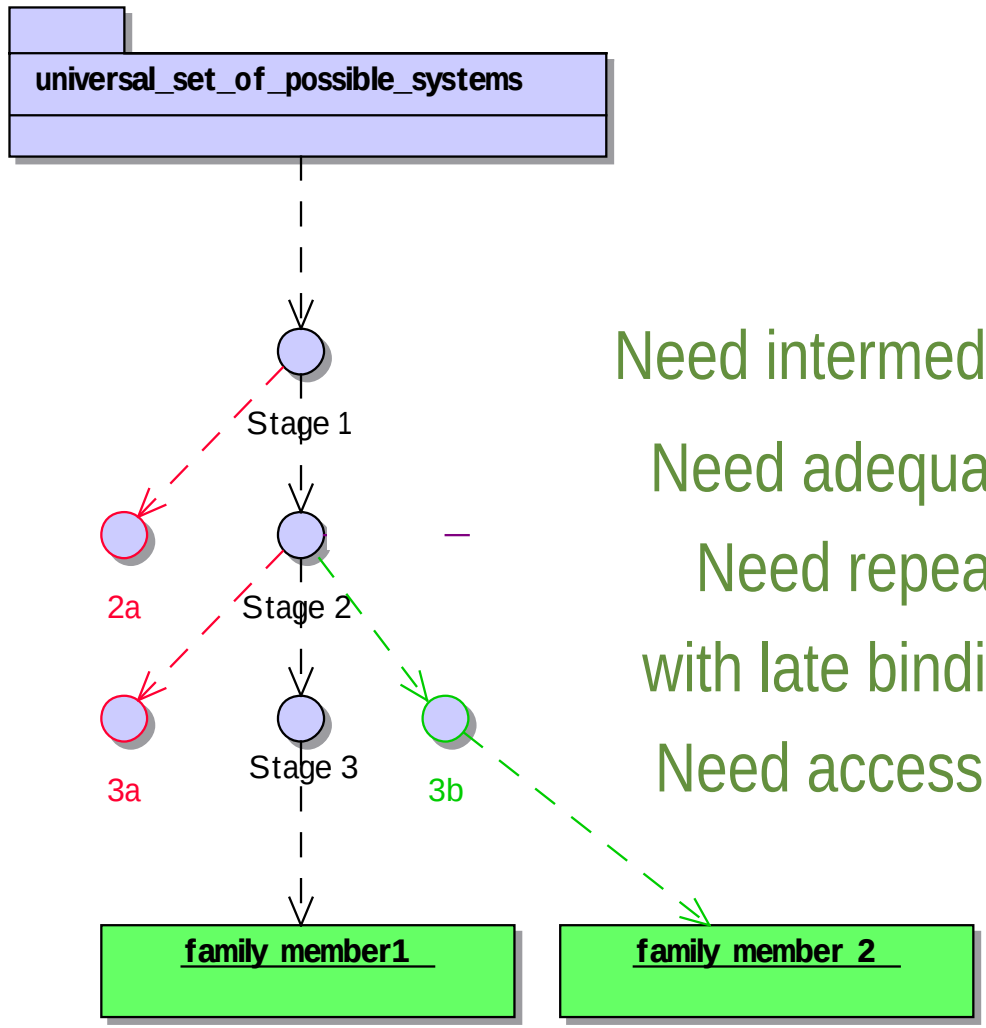



When the design had not yet been bound to the target OS!

kfrank@borland.com

Borland

# Need to find the right intermediate stage later!




Need intermediate workproducts

Need adequate specifications

Need repeatable methods

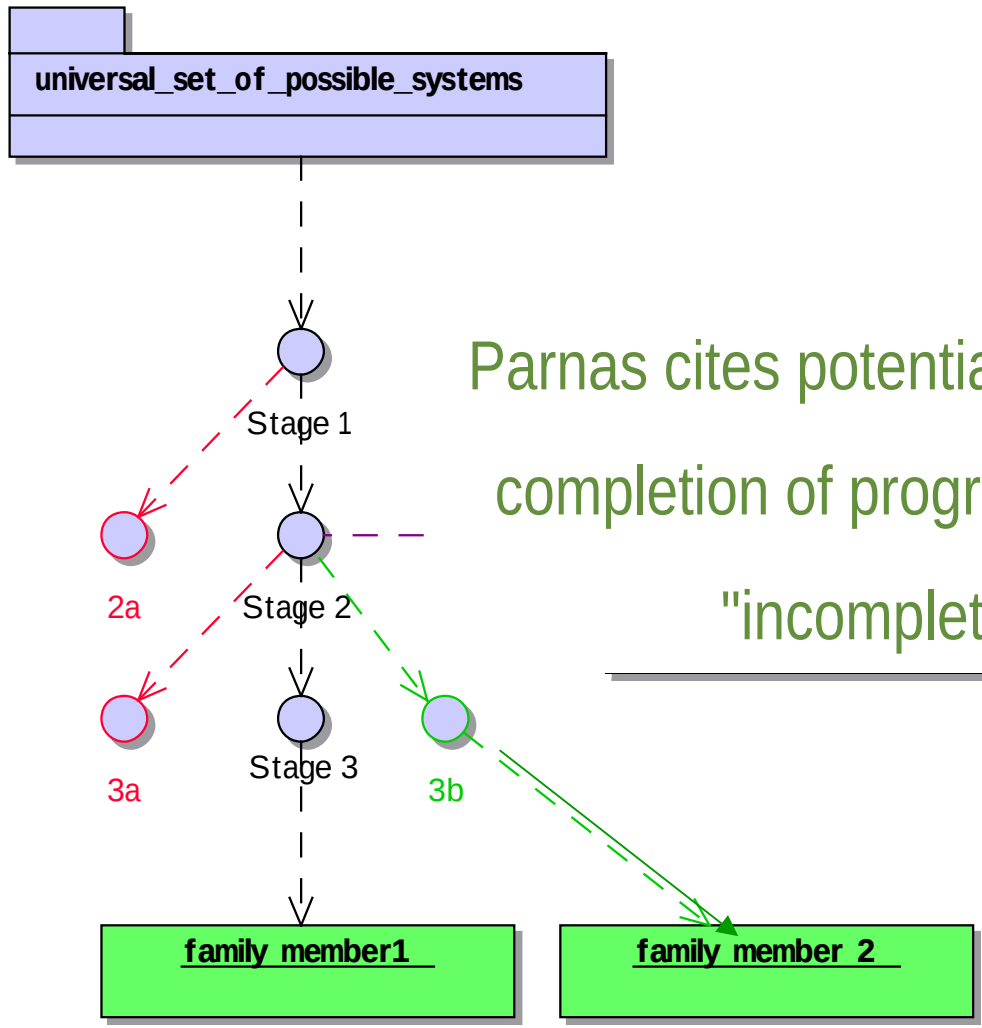
with late binding to technology

Need access to prior projects

kfrank@borland.com

Borland

# Origins of Product Family and MDA Methods




Parnas cites potential of generators for completion of program from selected "incomplete" stage

kfrank@borland.com

Borland

# Up to 2000 and MDA Initiative in one step

## The "Black and White Book"

- *Generative Programming: Methods, Tools, and Applications*, Krzysztof Czarnecki and Ulrich Eisenecker, 2000
  - *Of course, much else intervened. Weiss and Lau, etc.*
  - *Interest in Reuse fed into Domain Analysis at high level*
  - *At lower levels, Reuse fed into Coplien's idioms*
  - *And in between, Design Reuse fed into GoF Patterns*
  - *And product line engineering (Analysis of Points of Variation)*
  - *Dissatisfaction with wordy specs fed into UML*
  - *Unhappiness with specs fed specification language research*


kfrank@borland.com

# MDA and Software Factory (SF) Approaches

These two share the same foundation

- The thread runs from Parnas to Czarnecki & Eisenecker
- The thread runs thru the reuse and domain engineering movements
- Czarnecki served as consultant and reviewer for the *Software Factory*(SF) book by Greenfield and Short, Microsoft's response to MDA.


kfrank@borland.com

# How do these related approaches differ?

One clear difference lies in SF's negative attitude towards UML

Domain Specific Languages (DSLs), restricted to one *Problem Domain* and earlier known as "Little Languages," are the focus in SF instead of UML models.

Interesting new idea here is that a DSL is itself a medium for encoding domain knowledge and reusing it across projects.

*Moving problem domain knowledge into the meta-level!*

*SF approach is consistent with MDA if one uses MOF to define the DSL.*

Another difference: the cameo role of Zachmann framework in Software Factories is not in the MDA vision.

Borland®

kfrank@borland.com

# First Lesson:

*MDA is applicable to product families generally, not restricted to the case where the point of variability is middleware, programming language, or operating system.*

*(the potential is even greater than the PIM/PSM view promises)*



kfrank@borland.com

# Pre-MDA feature

Pattern Tools

- Select a straightforward analysis model in UML
- Select the implementation pattern you wish to apply
- Click to transform the selected model to conform to pattern

Code is modified to implement the pattern.

Accelerates development, but fails to be MDA

- Transforms PIM to PSM, so that sounds like part of MDA, but in doing so, looses the intermediate workproduct!

Code generation alone doesn't get you to MDA.

Borland®

kfrank@borland.com

# Second Lesson:

*A number of different capabilities have to come together to achieve MDA.*

*Just as the promise of this view goes beyond PIM and PSM, the problem for the tool vendor goes beyond providing a transformation tool.*

*Access across projects to reusable workproducts is one part.*

*To provide the necessary discipline, today it is most effective to build all the stages and capabilities into one tool that takes the application from cradle to grave.*

Borland®

kfrank@borland.com


Image copyright 2005 Karl Frank

# One example: Common Project

We have seen that one prerequisite for real MDA is access to intermediate workproducts from prior projects. Yet the tools in today's toolchains each create their own so-called projects, and have no support for the project manager's view of what a project is.

So here is a capability required of any true MDA tool:

Description

Associate individual tool projects with a wider Project context.

Key Features

- Assign any number of "projects" in RM to Project.
- Assign any number of VC "projects" to a Project.
- When user performs SSO, she can select a Project and the appropriate projects or views.
- VC projects and views can be tagged to assist in selection of appropriate tools.
- The environment caters to project manager's view of project. Current tools cater to developers' views.

Borland®

kfrank@borland.com

# Third Lesson:

*Main-stream MDA products of today are really **ARAD** (Architected Rapid Development) more than **MDA**.*

*Used for one-off creation of a program on one platform from one model, purpose-made for this one solution*

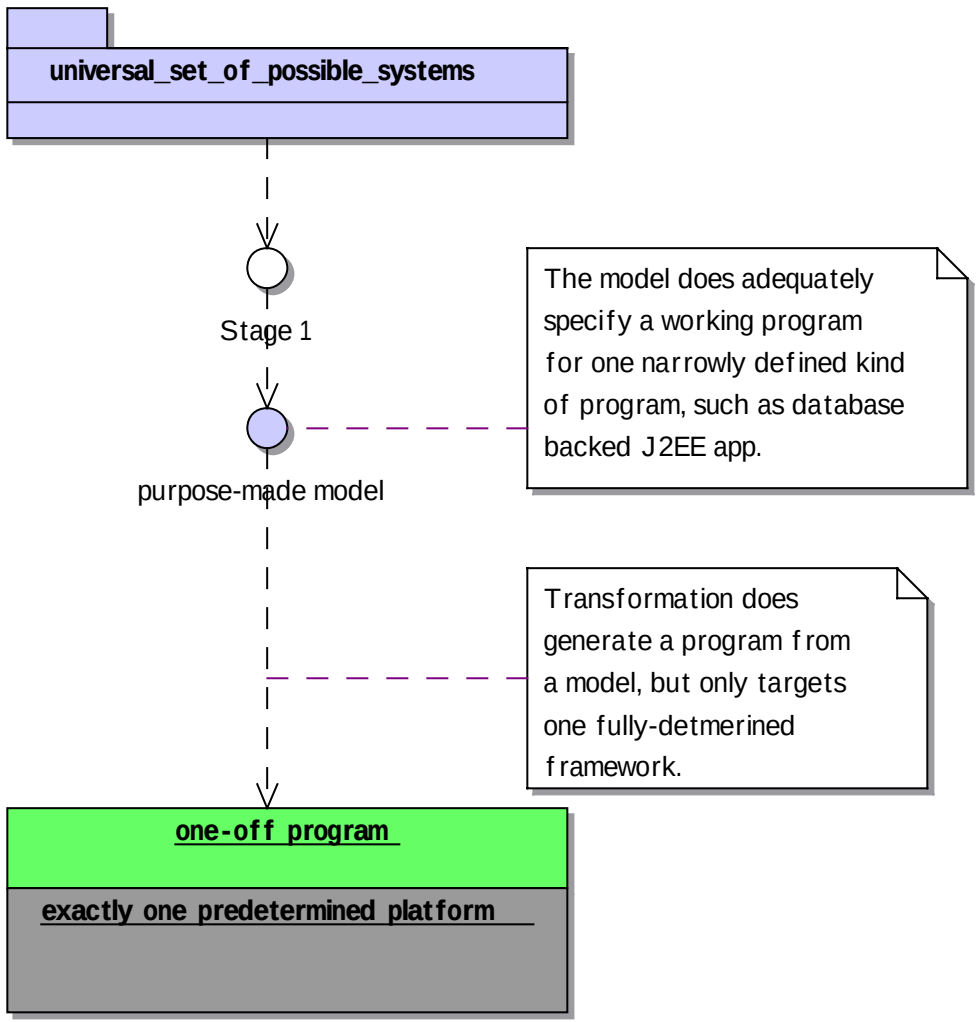
*Examples: IBM RapidDeveloper®, Compuware's Optimal-J®, Borland's Delphi ECO II®*





kfrank@borland.com

# Typical current achievement:




kfrank@borland.com

Borland

# *For adequate models, we need more than we have:*

Better model tests and audits

- OCL was needed, and has proven to make better specs

But more is needed:

- Consider this example.

Context: Bank developing new system to serve multiple product lines. Checking and Savings Products are in competition.

Suppose competing managers get rules defined as follows.

- no checking account without a savings account
- no savings account without a checking account

Borland®

kfrank@borland.com

# An ARAD tool for the VS framework/Delphi

Delphi™ 2005
Architect

Case Study | Product News | Events | Training | Buy Now

Products

Delphi
Architect
Enterprise
Professional

## ▸ ECO II

**Accelerating the development lifecycle with Borland® ECO™ II Object/Relational-mapper and ECO Space technology**

New Borland® Enterprise Core Objects II (ECO™ II) rapid design-driven architecture for .NET speeds development, improves quality, and increases the maintainability of complex Web Services, ASP.NET, and WinForm applications. Architects can use the model-powered ECO Object/Relational Mapping technology to generate or map to several major enterprise-class relational databases, then later evolve the database based on changes in the design model. Build enterprise-level model-powered applications using the new ECO Synchronization Server, offering multiple synchronized ECO Space object caches for increased scalability and performance.

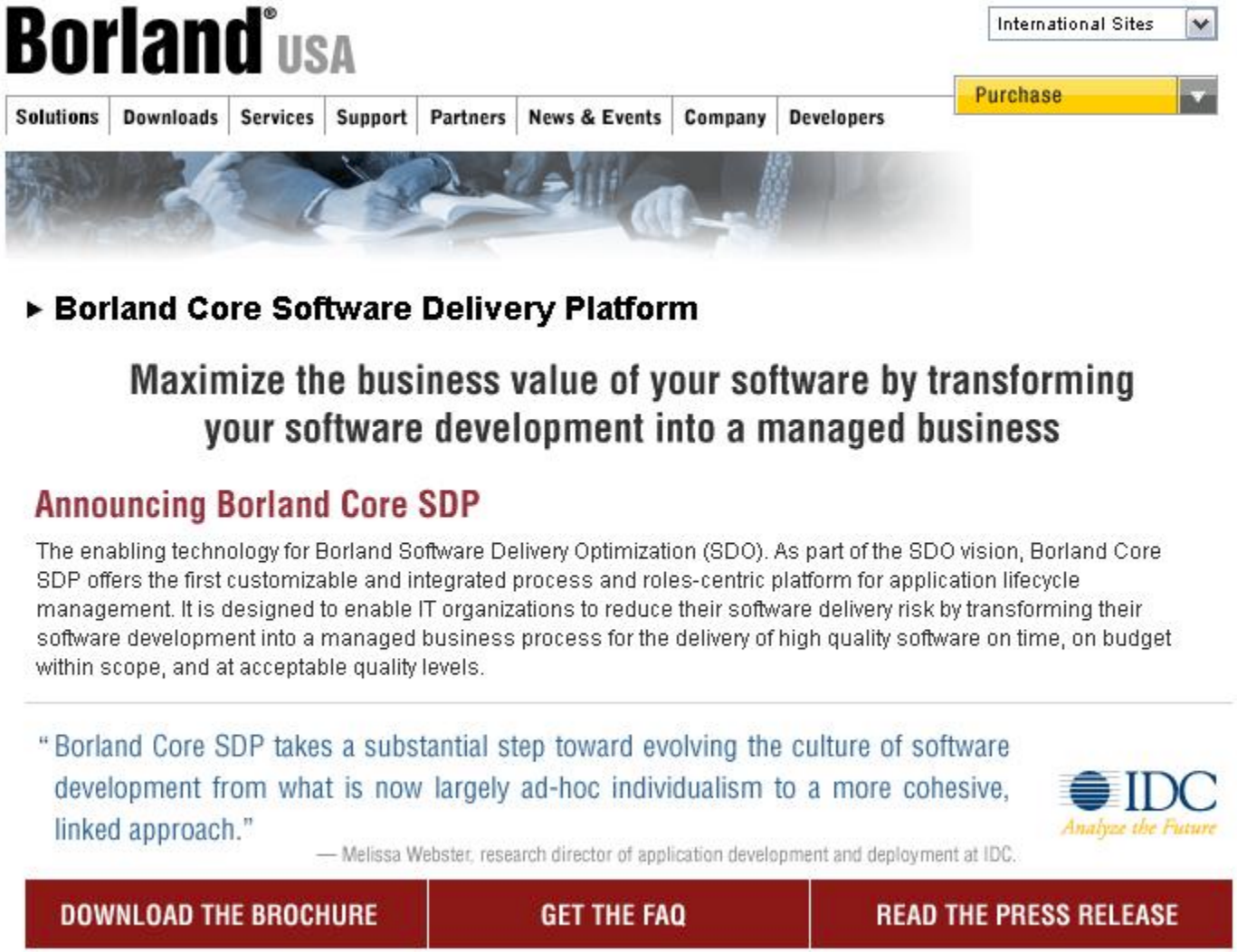
The following tutorials will take you step-by-step through building a variety of ECO applications.

- Tutorial 1: Creating your first ECO application
  This article demonstrates building a simple application using the Borland Enterprise Core Objects (ECO)™ technology in Delphi 2005 Architect.
- Tutorial 2: Working with Associations
  This article demonstrates accessing Borland ECO™ objects in code via handles, implementing relationships and using master/detail style user interfaces.
- Tutorial 3: Using ECO™ II with Databases
  This article demonstrates connecting an ECO™ application to a RDBMS and managing changes to the object space.
- Tutorial 4: Using an existing enterprise database with Borland® ECO™ II and Borland Delphi® 2005
  This paper provides step-by-step instructions on how to promote an existing relational database, including existing data, and how to provide a Unified Modeling Language™ (UML®) class diagram representation of this.

Borland®

kfrank@borland.com

# A Borland Strategic Initiative




Borland

kfrank@borland.com