The OPEN Group Architecture Framework
and
The OMG's Model Driven Architecture

Allan Kennedy

Kennedy Carter Ltd
www.kc.com

John Spencer

The OPEN Group
www.opengroup.org

The OPEN Group Architecture Framework
and
The OMG's Model Driven Architecture

Can a servant serve two masters?

ADM & MDA

A Crude Comparison of the two Belief Systems

- **TOGAF and ADM:**
- **A generic process oriented framework from which believers can derive enlightenment by identifying a Way (aka the right architecture for their organisation)**
- **Firm belief in what kind of deliverables the Way should produce**
- **Agnostic about the precise form of the deliverables and how to produce them**
- **Strong on ritual (lots of checklists and documentation)**
- **A relativistic faith, it is non-judgmental about particular religions (aka methods)**
- **A pantheistic religion - you can pick whatever belief system or article of faith is right for you from the infinite universe (aka the enterprise continuum)**
- **Borrows its mysticism from the whirling dervishes: believers iterate rapidly in circles**
- **A bit New Age - iconography clearly influenced by crop-circles**

A Crude Comparison of the two Belief Systems

- **MDA:**
- **Evangelical about deliverables (you can have anything you like as long as it is a model)**
- **Strong, possibly dogmatic, still evolving theology of modelling**
- **Mystical attachment to precise, executable and translatable models**
- **Models attain eternal life through detachment from wordly platforms**
- **Charismatic - believers model in many languages: MOF, UML, CWM, EDOC, CORBA (and more on the way)**
- **A very Rational religion - believers are always debating what it all means**

But Why TOGAF AND MDA?

- **The usual reasons - *Faster, Better, Cheaper***
- ***Faster better cheaper* development processes that result in…**
- **…systems that can be ported readily to *faster better cheaper* platforms as they become established**

MDA: Some Key Themes

- **Three primary viewpoints for modelling**
  - ***The Computation Independent Model (CIM)***
    - Precise, oriented to stakeholders, uncommitted to specific algorithms
  - ***The Platform Independent Model (PIM)***
    - Precise, computationally complete (executable), uncommitted to any specific platform (portable), oriented to stakeholders (prototyping) and developers (a testable specification), a long life asset
  - ***The Platform Specific Model (PSM)***
    - Precise, complete, conforms to constraints of a specific (class of) platform(s), probably generated from a PIM, oriented to technology experts
- **Interoperability defined at the model level, not the middleware level**
- **Models are integrated in order to construct complete systems**
- **Models are derived from other models by (partially) automated transformation**
- **So, MDA promises to be rigorous but lightweight, -> extreme modelling**

TOGAF AND MDA: A Unifying Theme

- **TOGAF, and particularly the ADM, exposes and/or generates a considerable quantity of knowledge about the actual and planned architecture (at all four levels) of a system or system of systems**
- **MDA provides a set of languages for formalising that knowledge as stakeholder-friendly models that can be verified for correctness**
- **…and that can be leveraged to automate the production of downstream deliverables.**

F-16 Modular Mission Computer Application Software

Reducing Test and Integration Time with Model Driven Architecture and Executable UML: The Lockheed Martin Experience

Lauren E. Clark
Chief Engineer
F-16 Modular Mission Computer Software
Lockheed Martin Aeronautics Company

Bary D. Hogan
Methodology Lead
F-16 Modular Mission Computer Software
Lockheed Martin Aeronautics Company

Terry Ruthruff
Staff Specialist
Software Engineering Core
Lockheed Martin Aeronautics Company

Allan Kennedy
Managing Director
Kennedy Carter Limited

Background

- **F-16 originally designed as a lightweight air superiority fighter**
  - ***Jointly built by Belgium, Denmark, Netherlands, Norway, US***
- **First F-16A flew Dec 76, first operational Jan 1979**
  - ***More than 4,000 F-16s manufactured to date***
- **Modular Mission Computer (MMC) in post-July 1996 aircraft**
  - ***Also to be retrofitted to 1,000 earlier aircraft***
  - ***Replaces computers for Fire Control, Head-up display, Stores management***
  - ***Multiple MIPS R3000 64-bit CPUs, 60 MB memory, 155 MIPs***
  - ***30x throughput & memory at 60% weight, volume & power***

Agenda

- **The Platform**
- **Cross-Platform Compatibility: The Goal**
- **Implementing MDA with eXecutable UML:**
  - ***Platform Independent Modeling***
  - ***Platform Specific Mapping (Design Tagging)***
  - ***Automatic Code Generation***
- **Benefits derived from using MDA**

Basic Software Components

Application Software:

- High-level software that is unique to the application(s) for which the embedded computer (i.e. subsystem) exists
- 80-90% of the total software (in terms of long-term development cost)

Software Execution Platform:

- Low-level software, the purpose of which is to provide services that allow the Application Software to run on the hardware

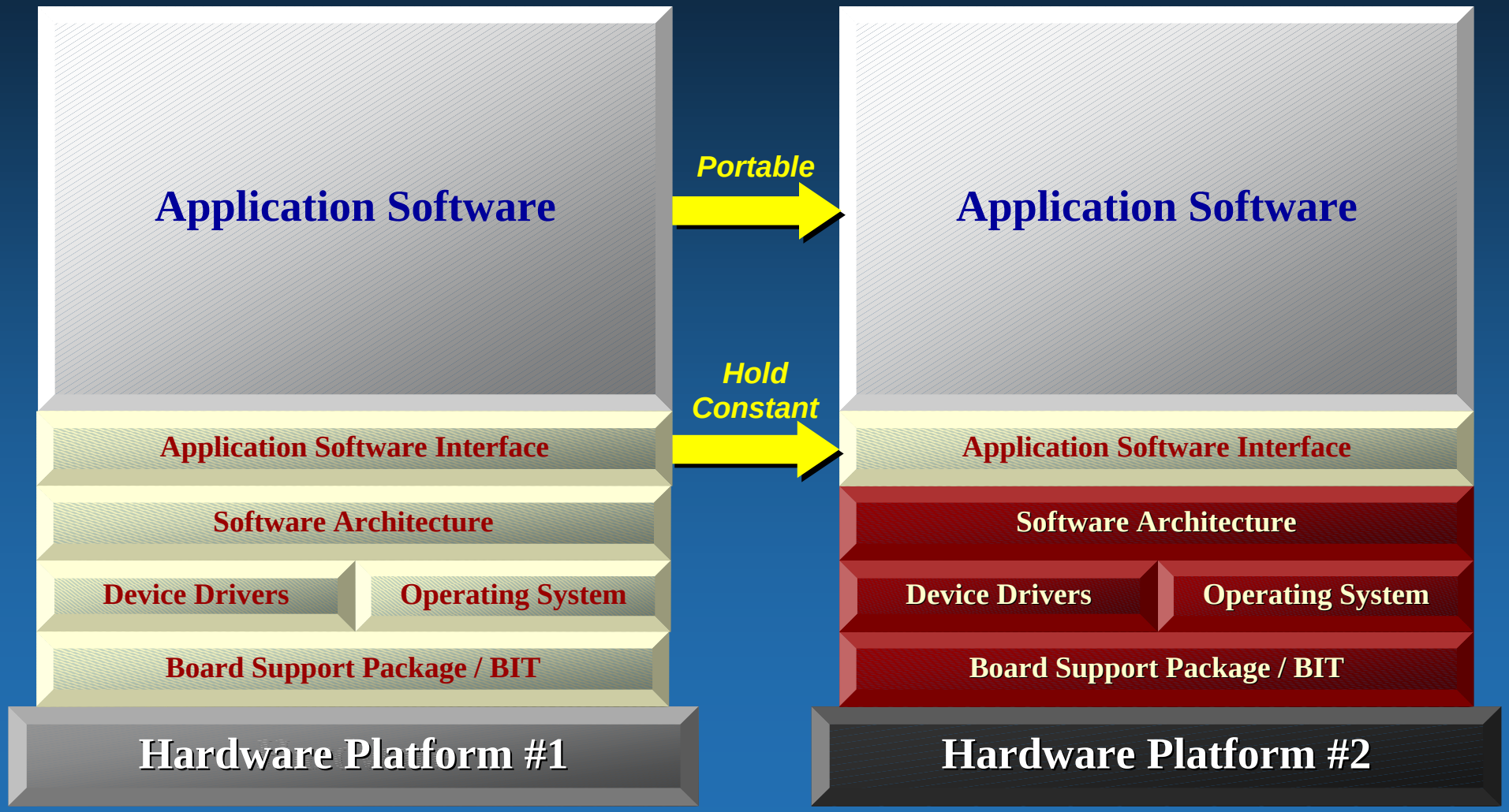
Software Architecture

Software Architecture:

- Low-level software providing the framework within which the Application Software executes
- Provides execution control, data / message management, error handling, and various support services to the Application Software
- Assumes a particular Application Software language
- Unique to the hardware; but, since it must support all requirements levied by the Application Software, is not delivered with the hardware

Cross-Platform Compatibility: The Usual Approach

Maintain a constant Application Software Interface

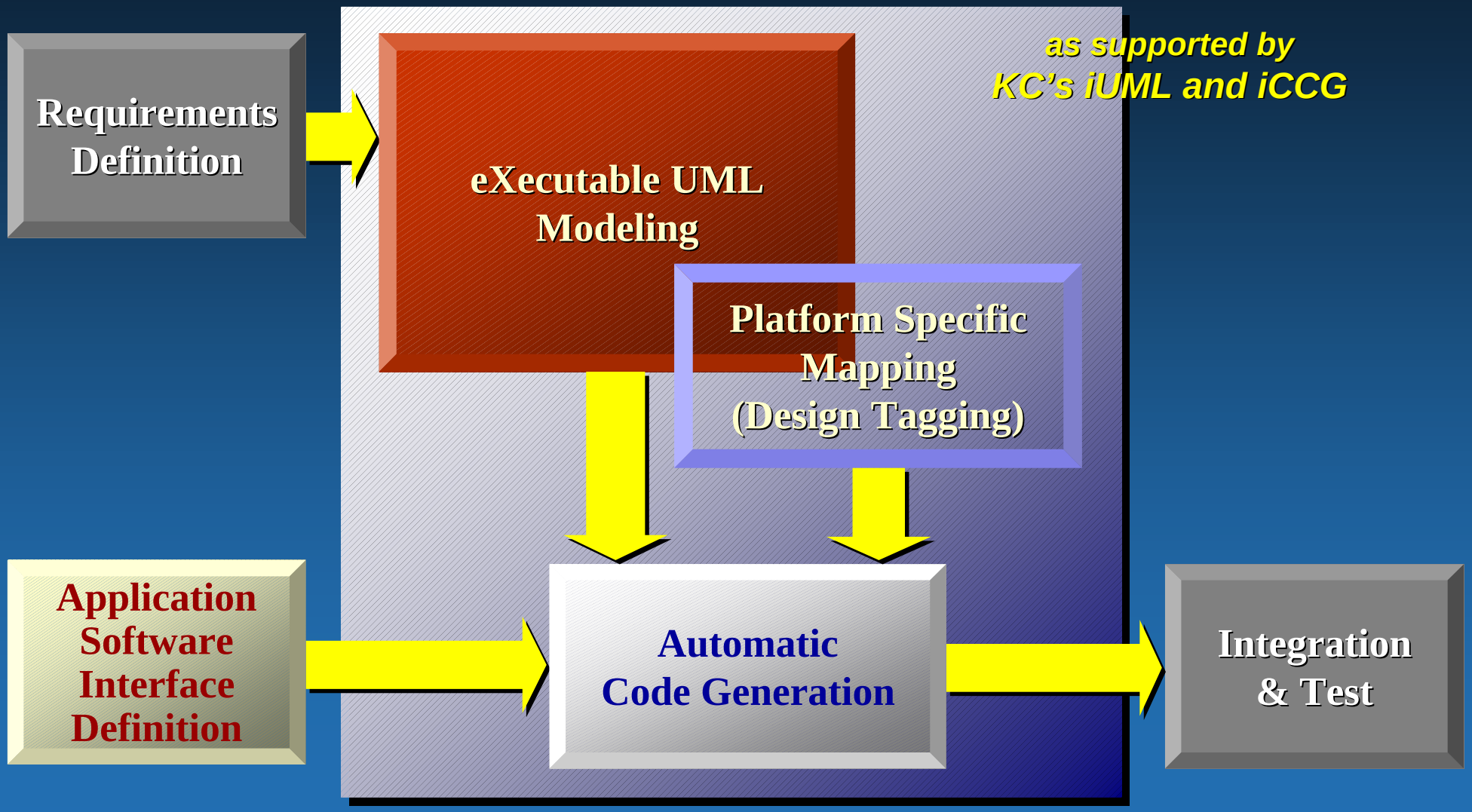


Cross-Platform Compatibility: The Goal

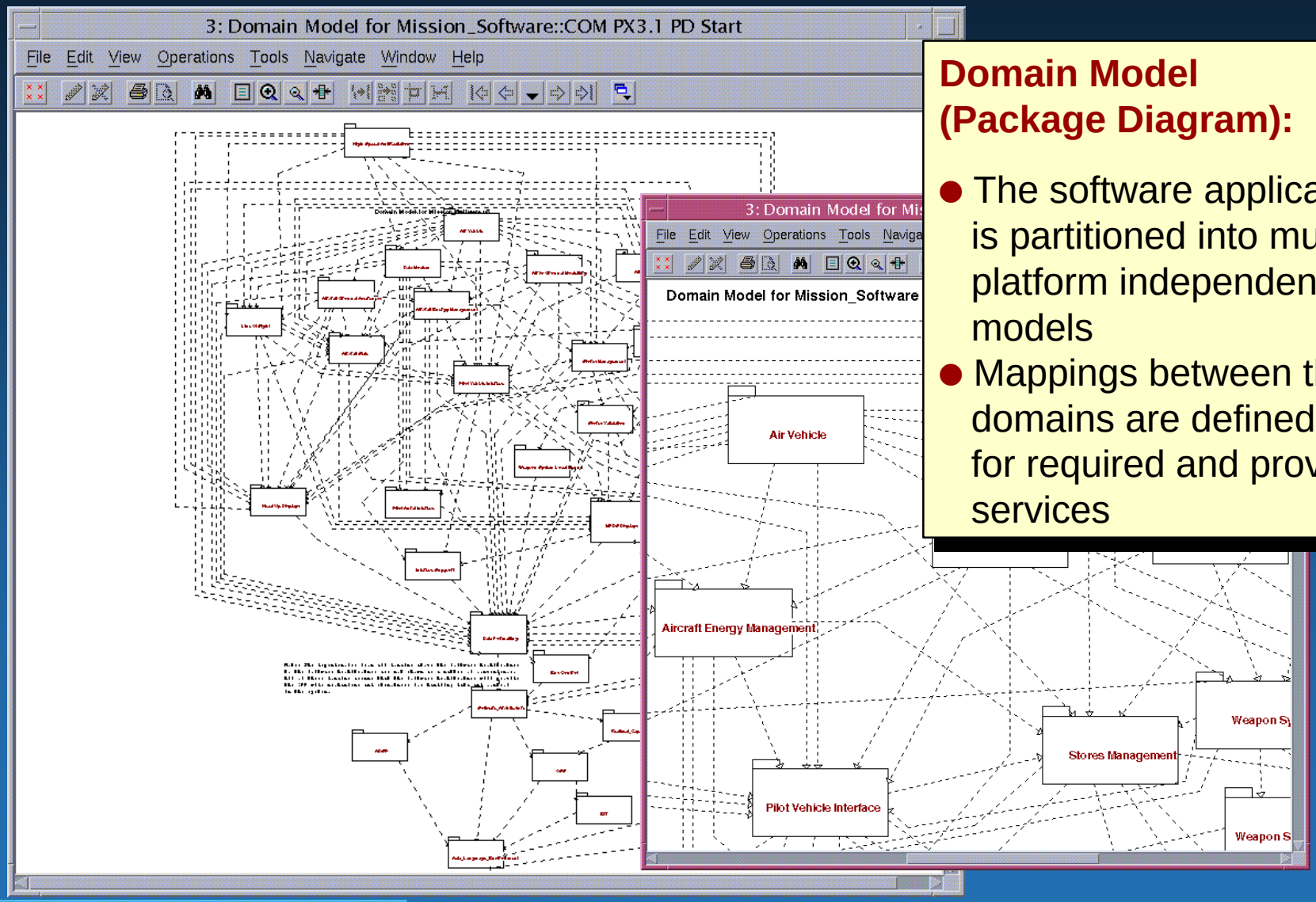
The goal should be to provide cross-platform compatibility of Application Software despite <u>any</u> Implementation, or platform specific, changes:

that is, changes to the Hardware Platform, the Software Execution Platform, or the Application Software Interface

MDA with eXecutable UML: Development Process



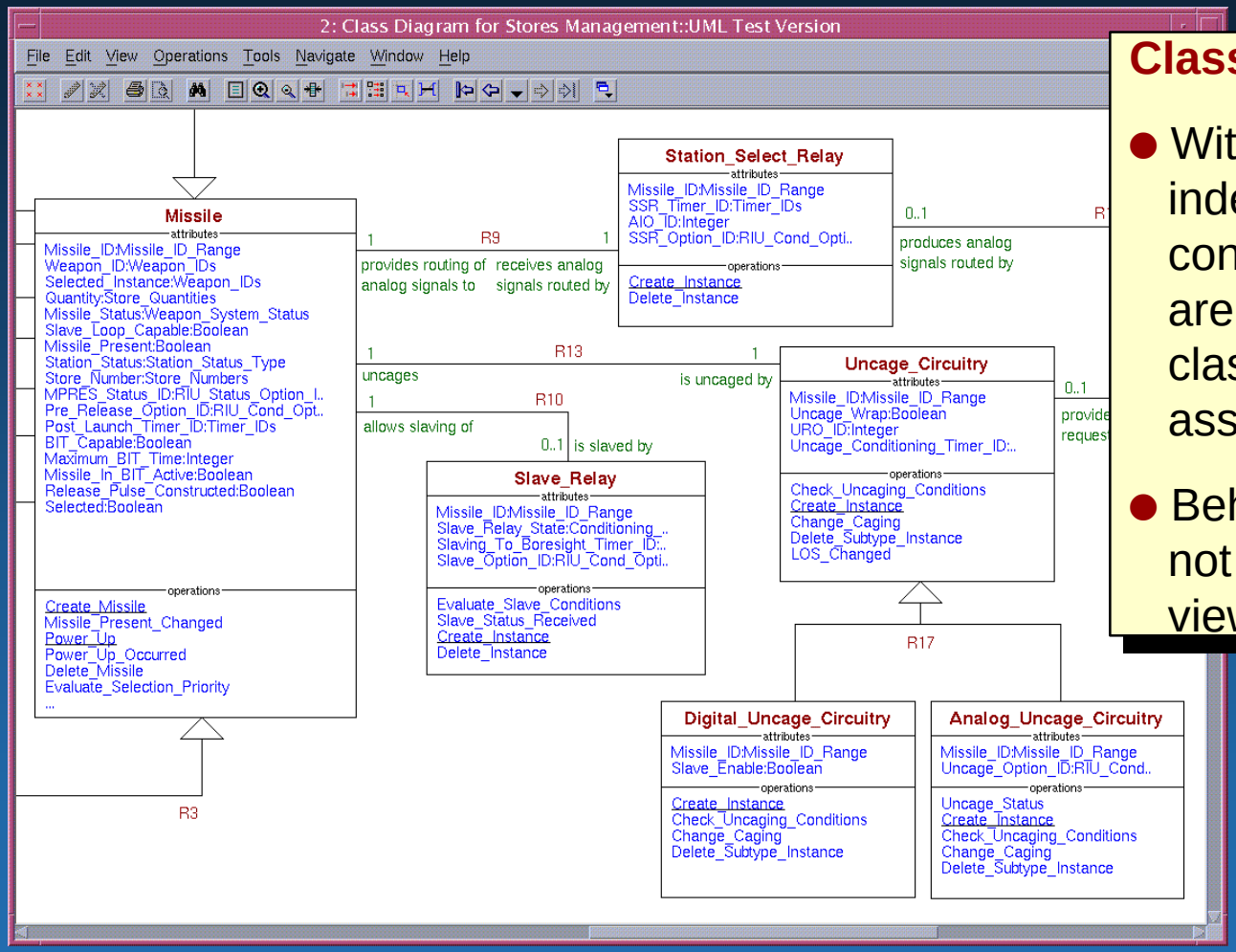
eXecutable UML Modeling: Domain Model



Domain Model (Package Diagram):

- The software application space is partitioned into multiple platform independent domain models
- Mappings between the domains are defined as contracts for required and provided services

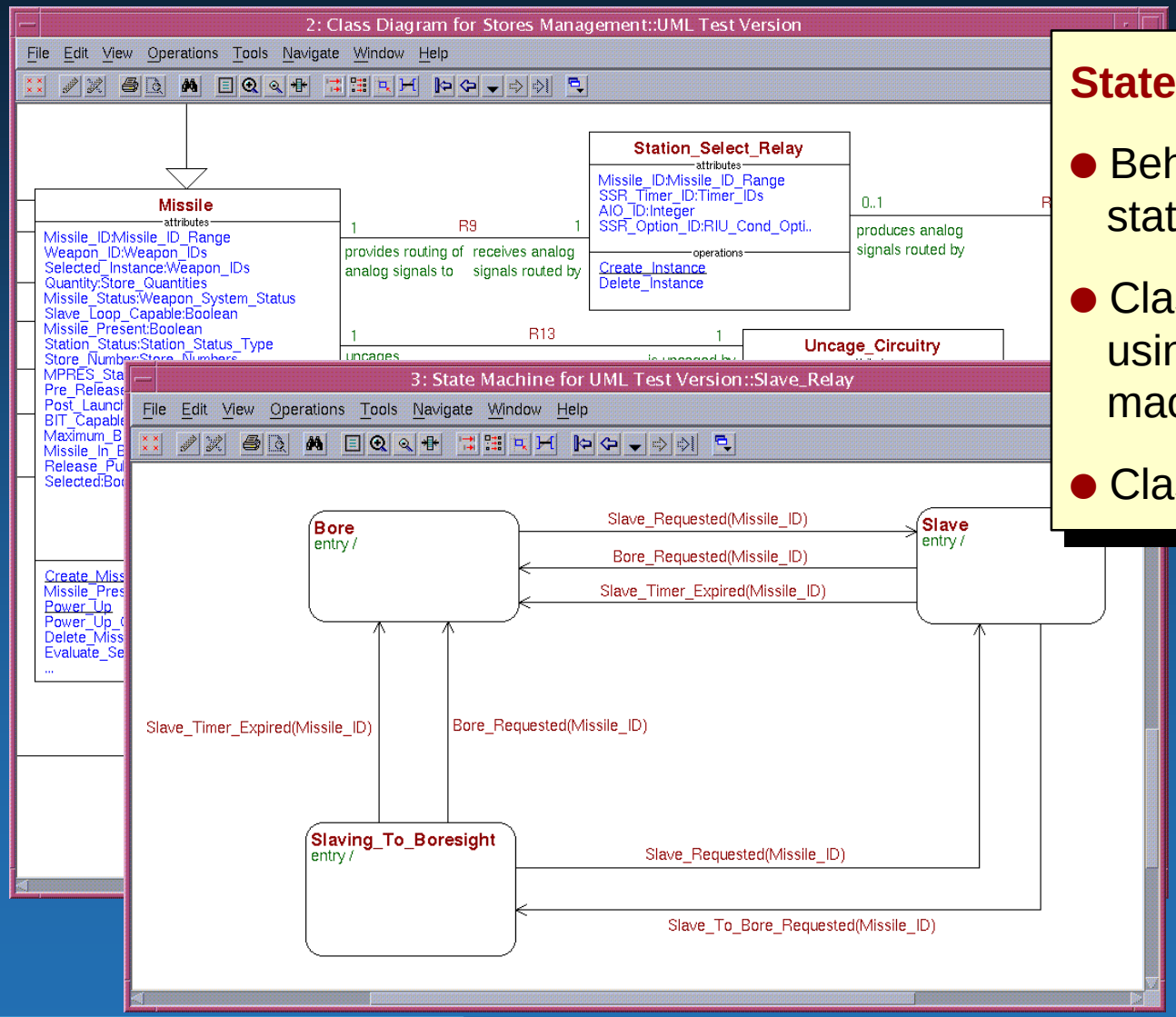
eXecutable UML Modeling: Class Diagrams



Class Diagrams:

- Within each platform independent domain model, conceptual entities are modeled first: classes,attributes, and associations are abstracted
- Behavior, though considered, is not modeled explicitly in this view

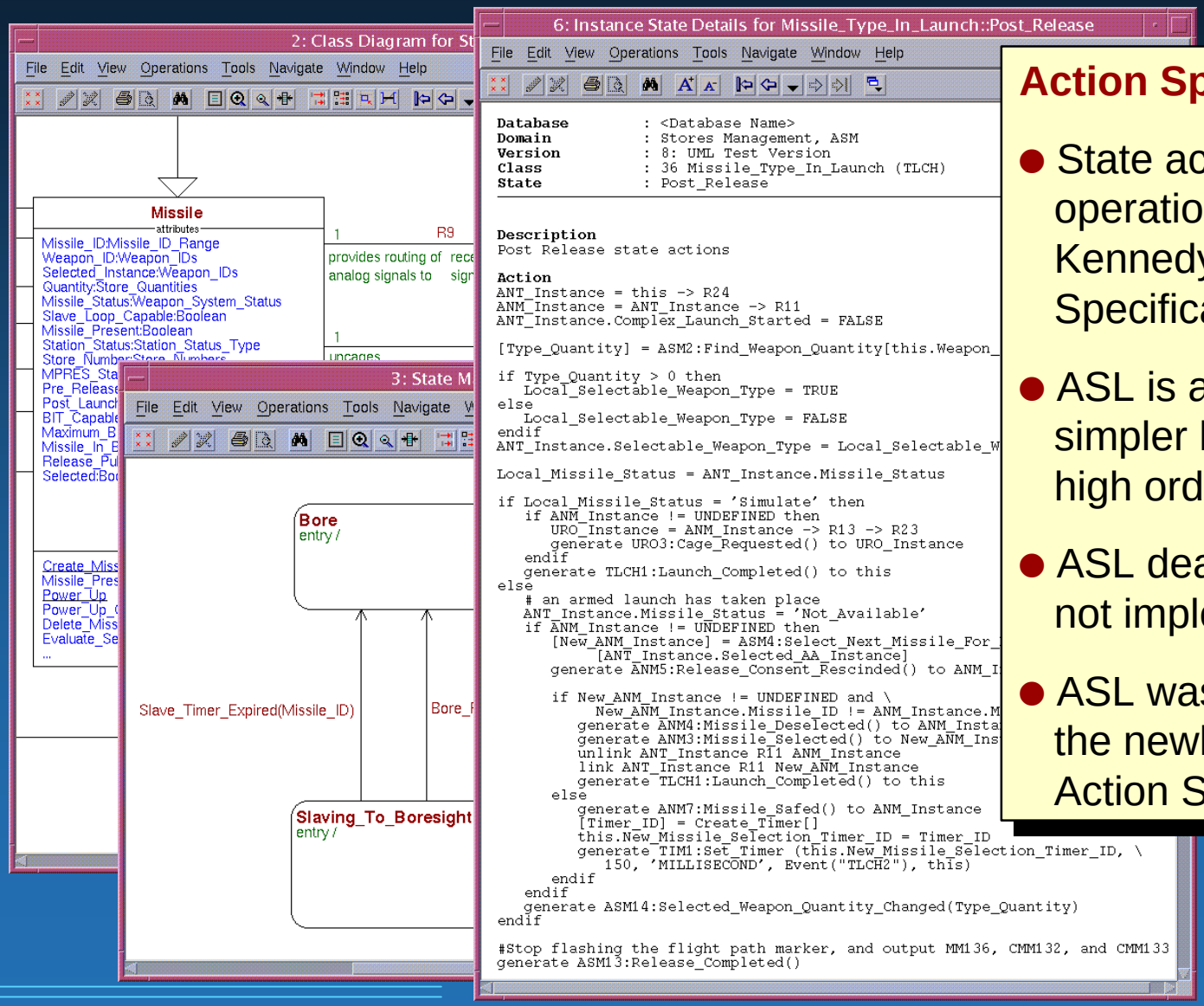
eXecutable UML Modeling: State Charts



State Charts:

- Behavior is formalized during state modeling
- Class lifecycles are modeled using signal-driven state machines
- Class operations are defined

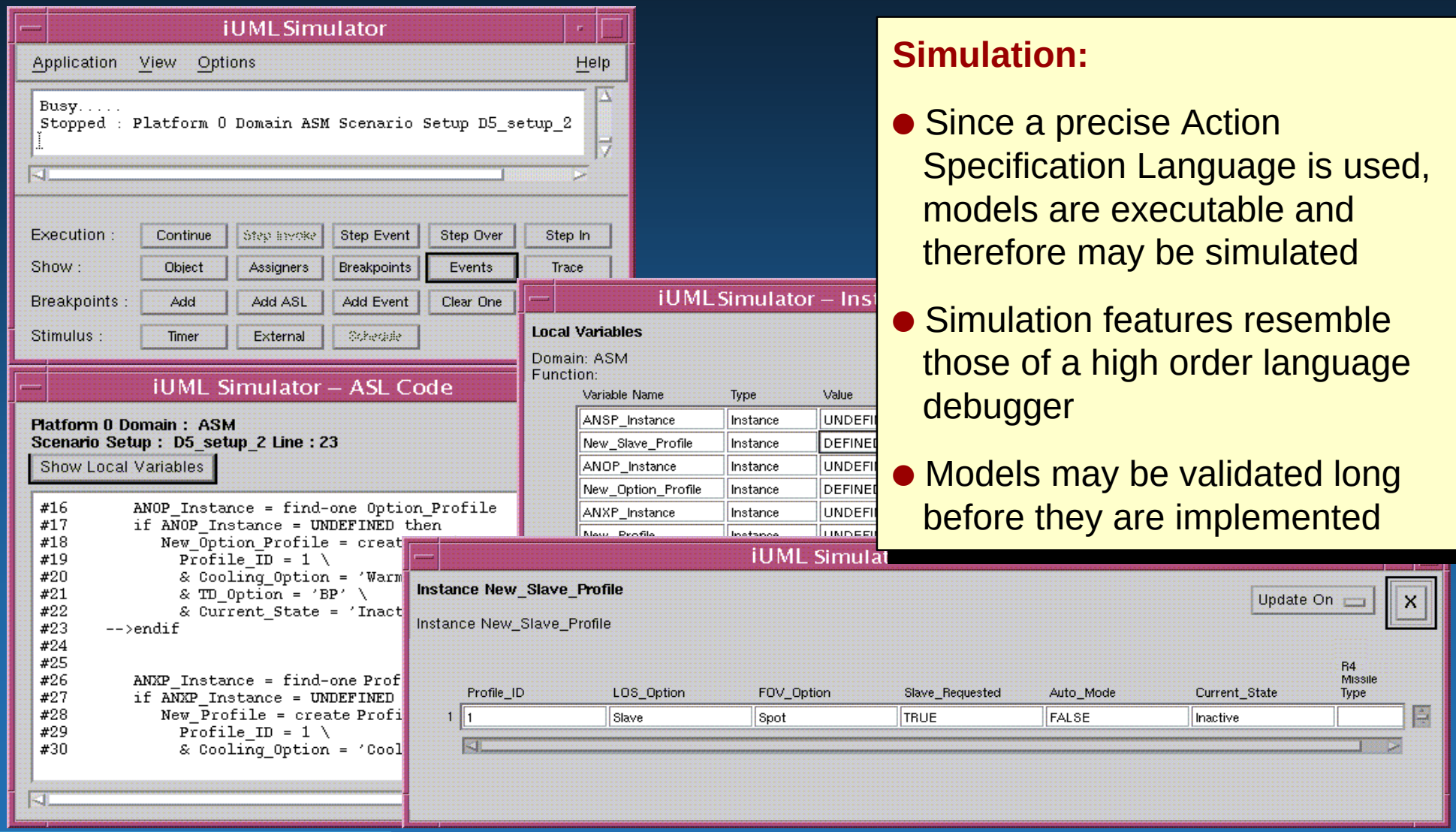
eXecutable UML Modeling: Action Language



Action Specification Language:

- State actions and class operations are specified using Kennedy Carter's Action Specification Language (ASL)
- ASL is a higher order and much simpler language than a typical high order language (e.g. C++)
- ASL deals with UML concepts, not implementation concepts
- ASL was a major influence on the newly adopted Precise Action Semantics for the UML

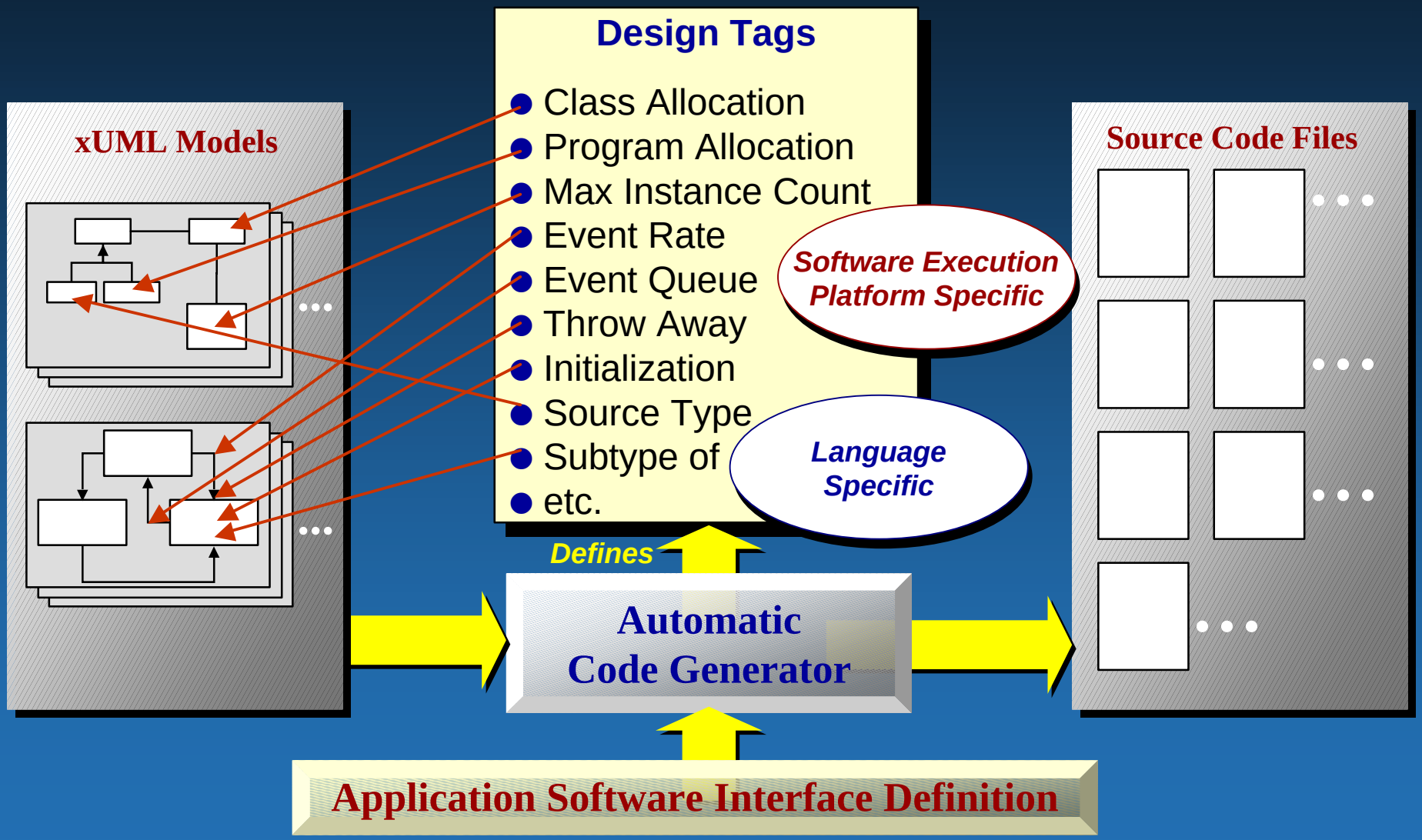
eXecutable UML Modeling: Simulation



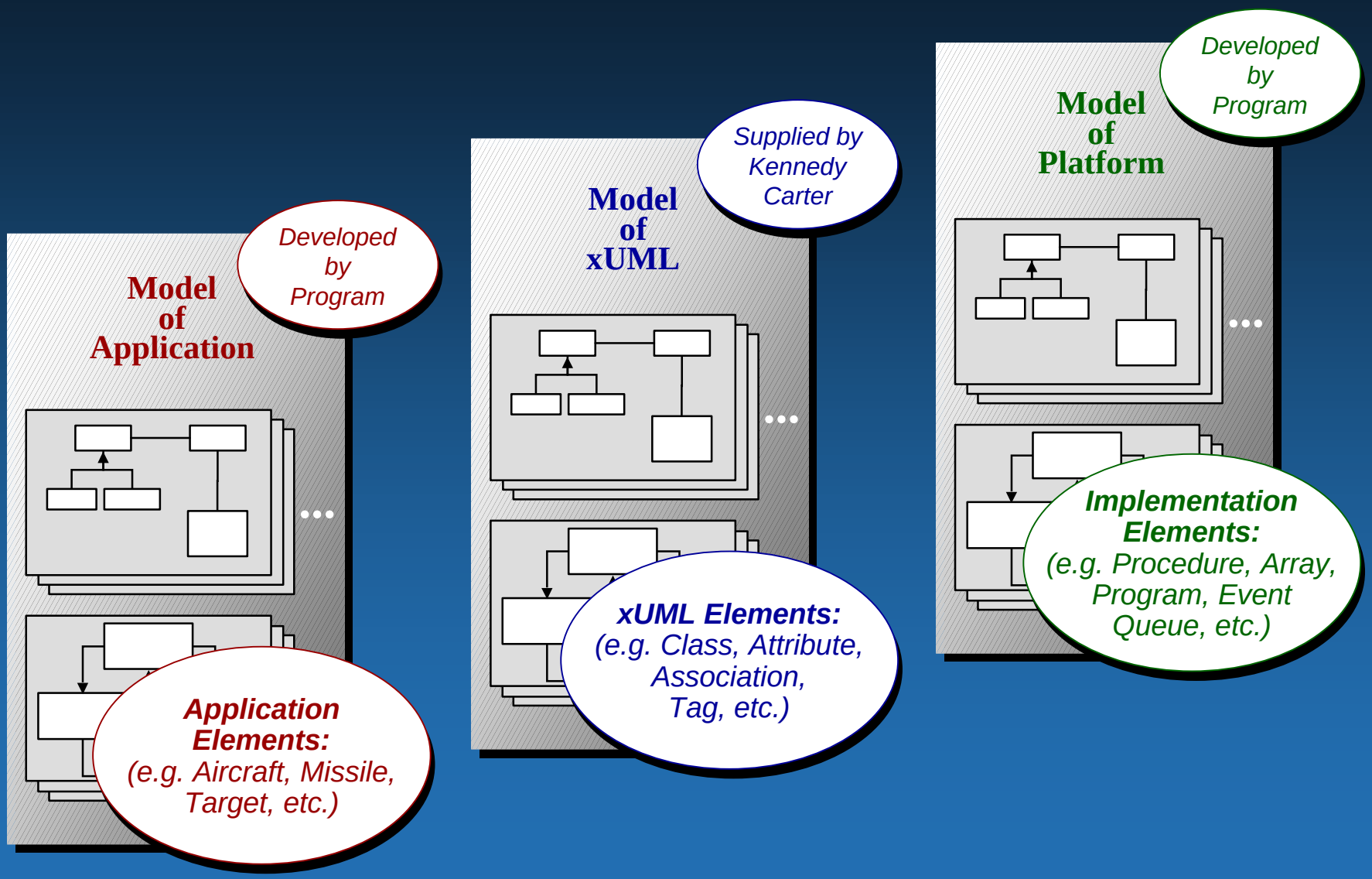
Simulation:

- Since a precise Action Specification Language is used, models are executable and therefore may be simulated
- Simulation features resemble those of a high order language debugger
- Models may be validated long before they are implemented

Design Tagging: Specifying the PIM to PSM Mapping

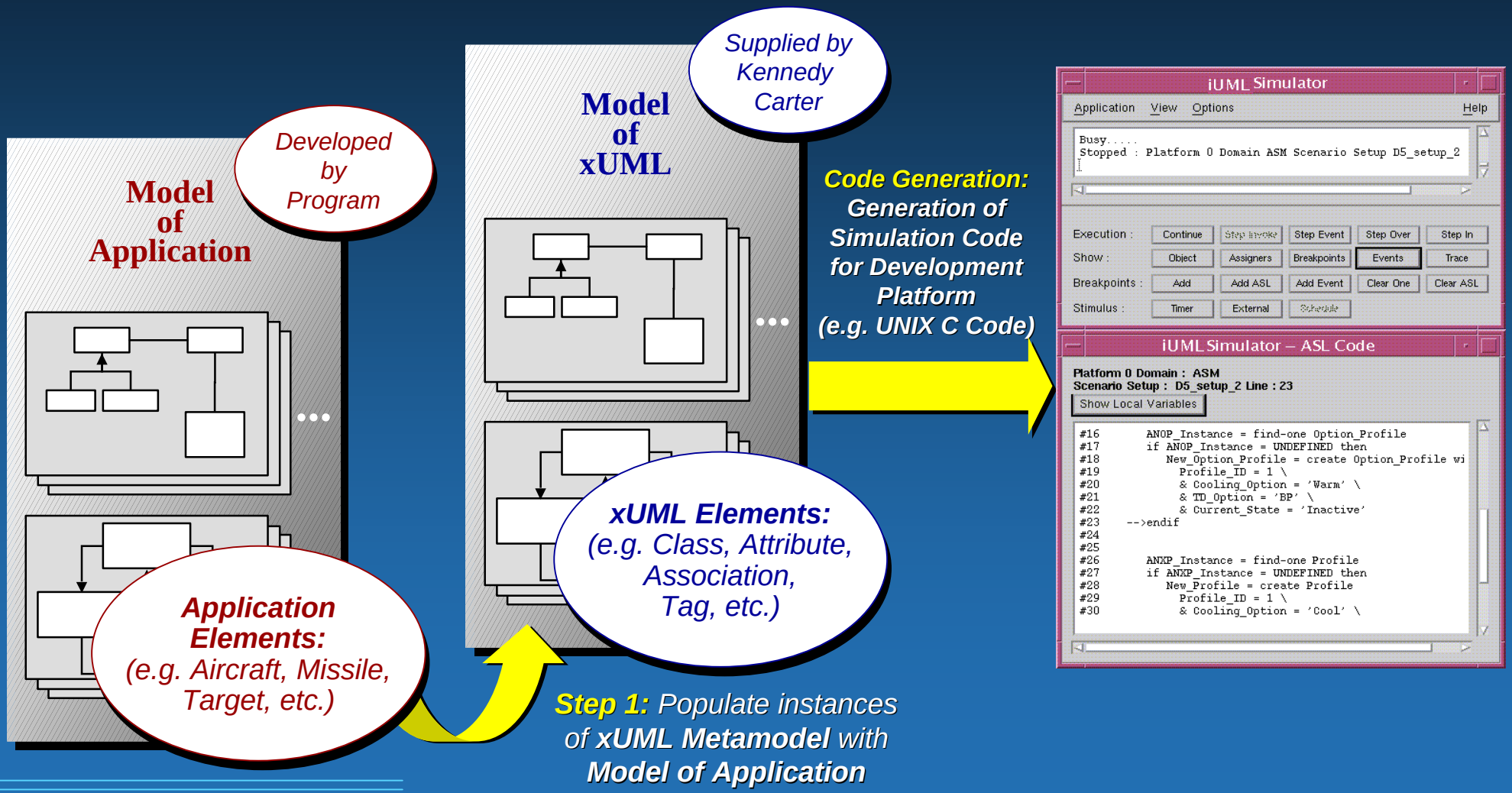


MDA: Automatic Code Generation

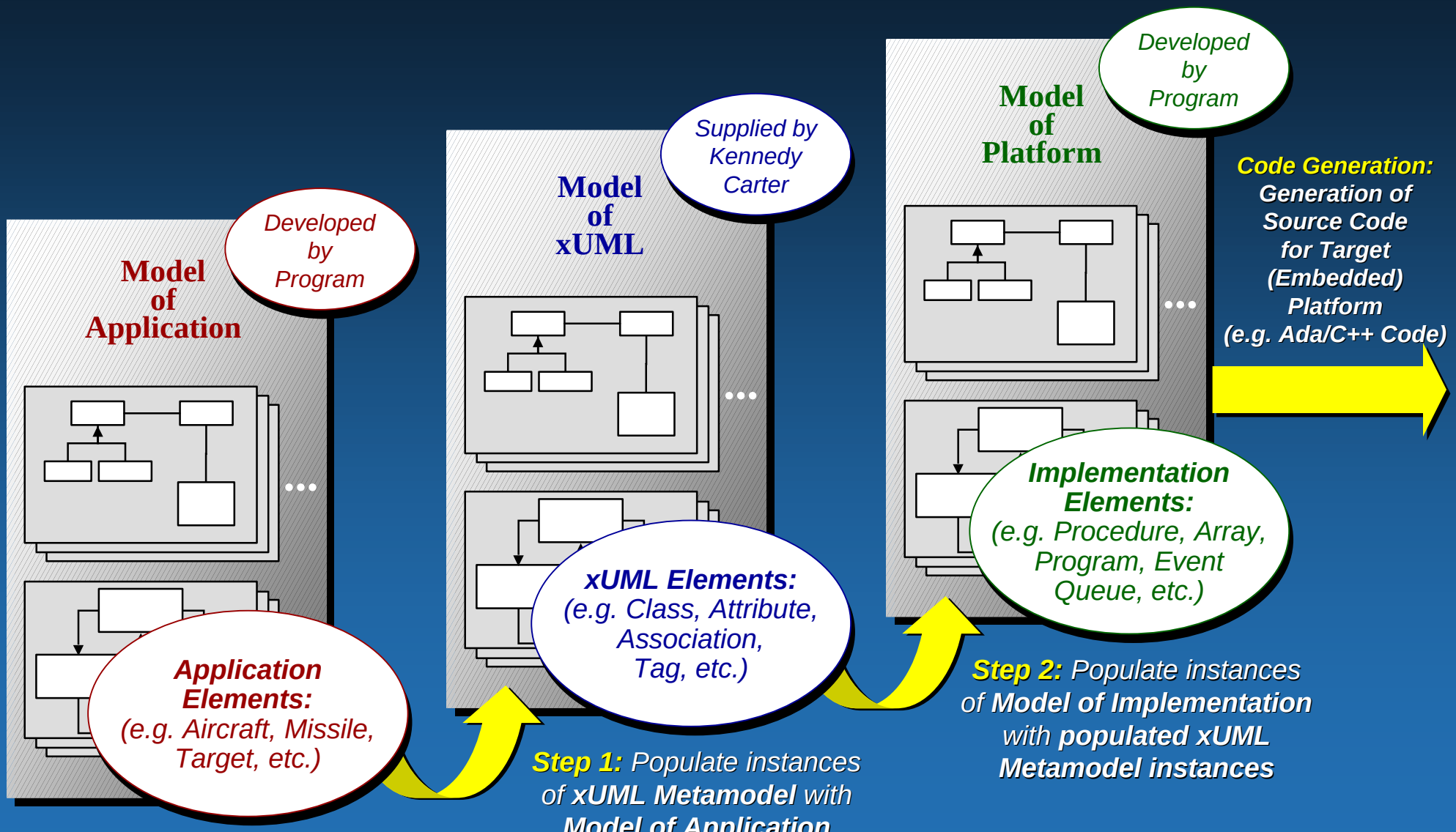


Automatic Code Generation: Simulation Code

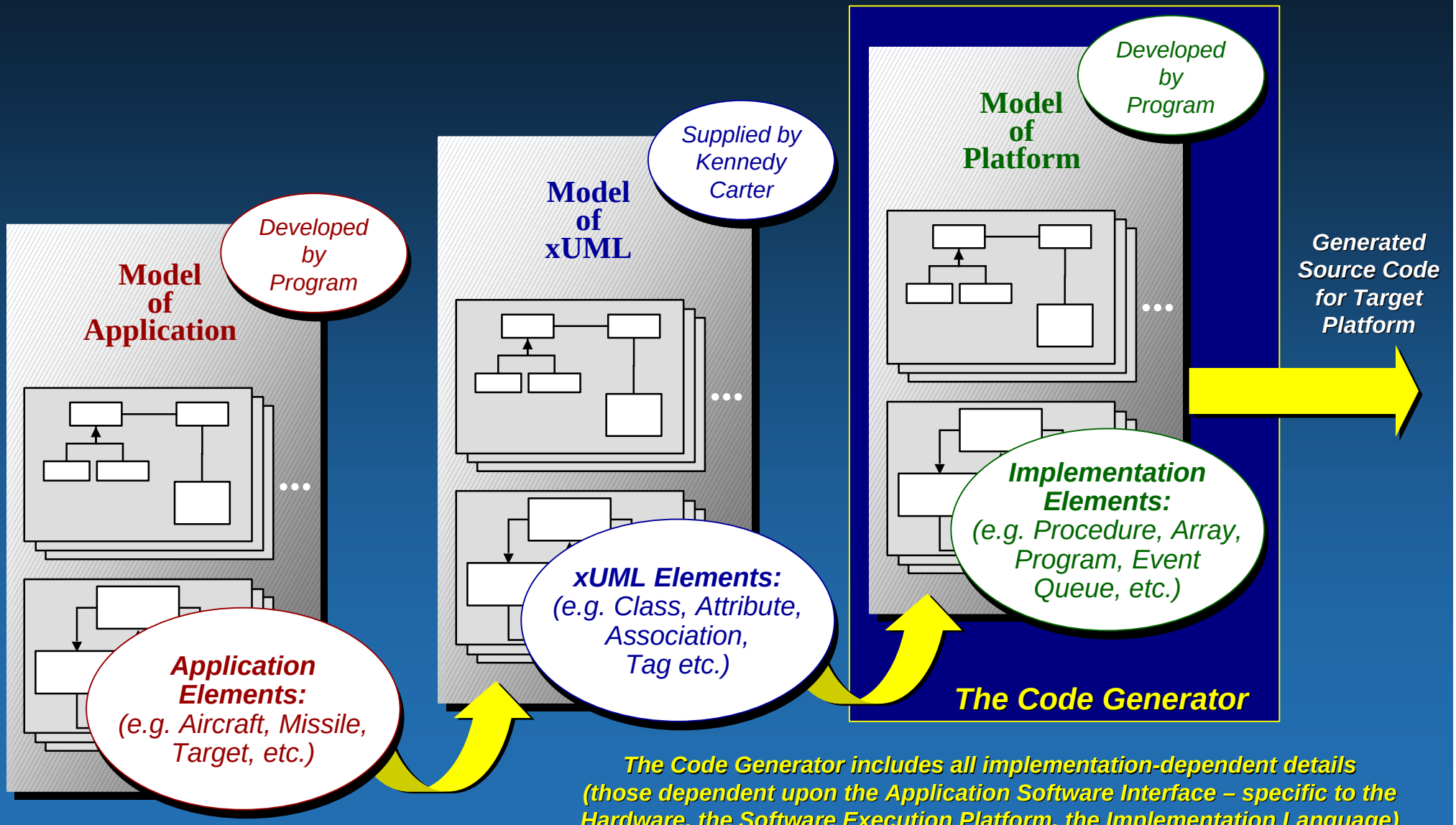
When we say that "xUML models are executable" we mean that "executable code can be automatically generated from them"



Automatic Code Generation: Target Code

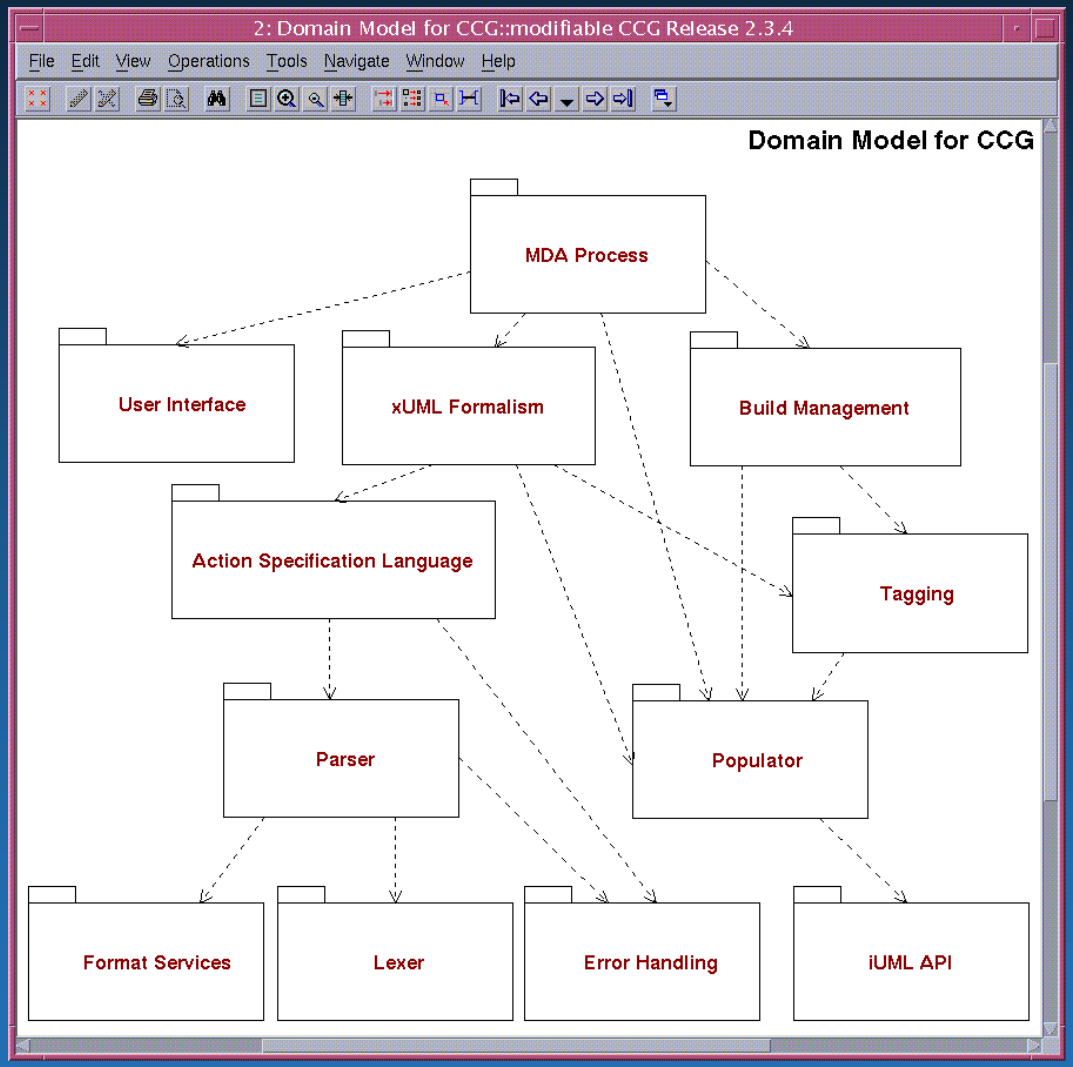


Automatic Code Generation: The Code Generator



The Code Generator includes all implementation-dependent details (those dependent upon the Application Software Interface – specific to the Hardware, the Software Execution Platform, the Implementation Language)

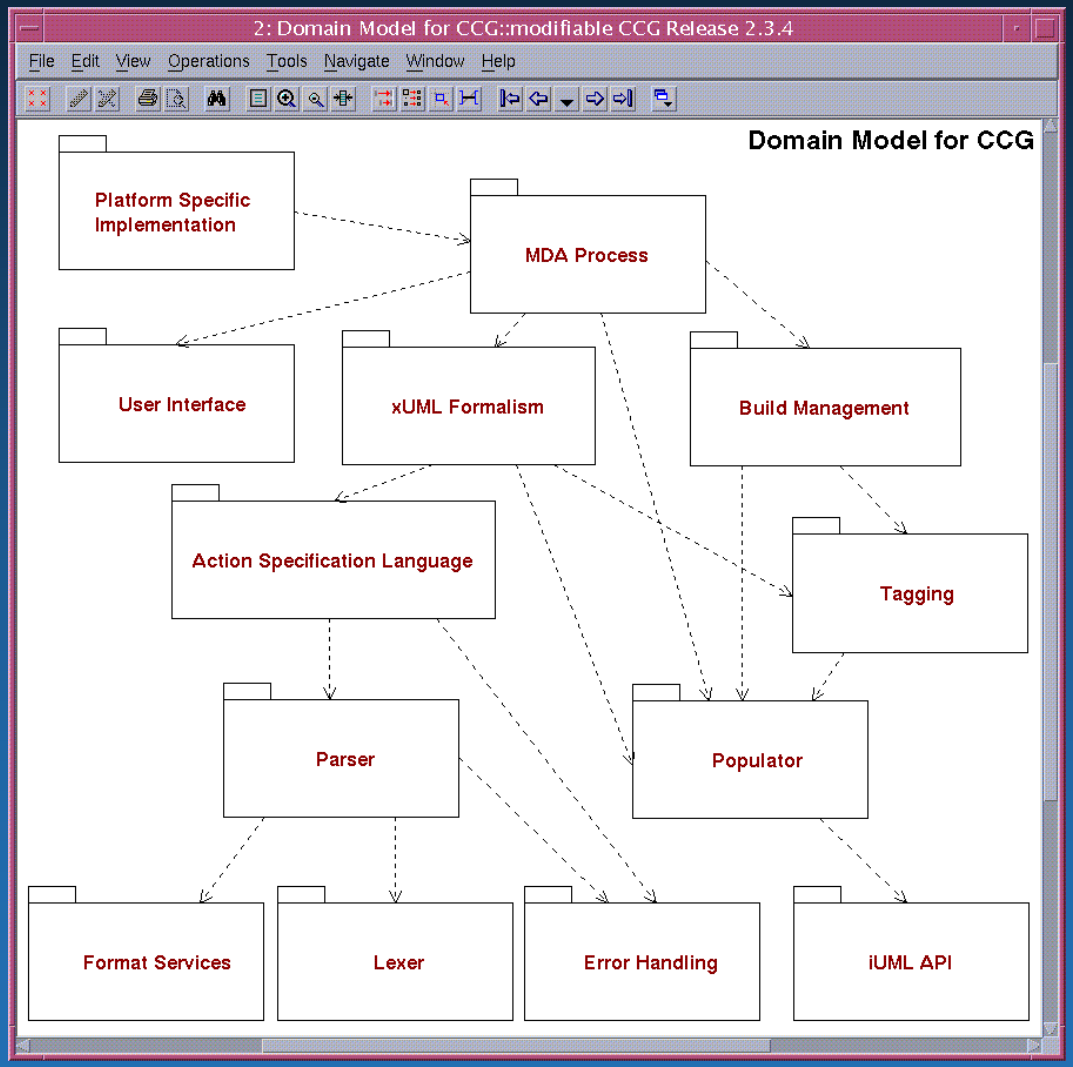
Automatic Code Generation: Code Generator Development



Configurable Code Generator:

- Code Generator is developed using the same eXecutable MDA strategy
- Kennedy Carter supplies a set of xUML models (known as the Configurable Code Generator) that serve as a generic translation framework

Automatic Code Generation: Code Generator Development



Code Generator Development:

- The Configurable Code Generator (iCCG) may be adapted to the meet the requirements of any Platform Specific Implementation (i.e. of any Application Software Interface)
- Code Generator and Application Software development may be performed concurrently

Automatic Code Generation: Summary

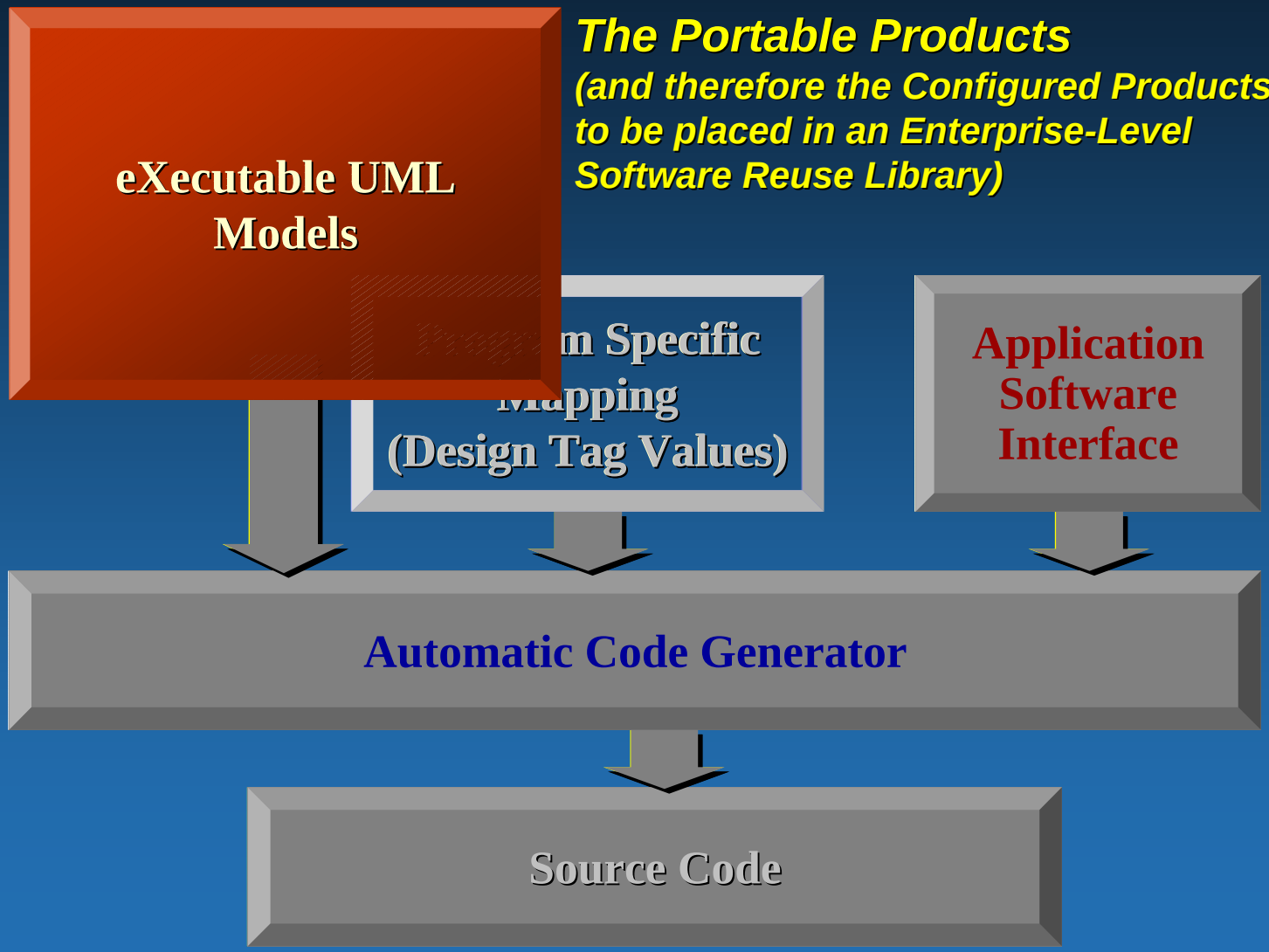
Automatic Code Generation

- **Automatic code generation is simply an extension of the code generation technique used for simulation of the eXecutable UML models on the development platform, this extension being for the target (embedded) platform**

- **The code generator is developed within the same environment as the application software using the same eXecutable MDA strategy**
  - ***Development cost: 1-2 architects***

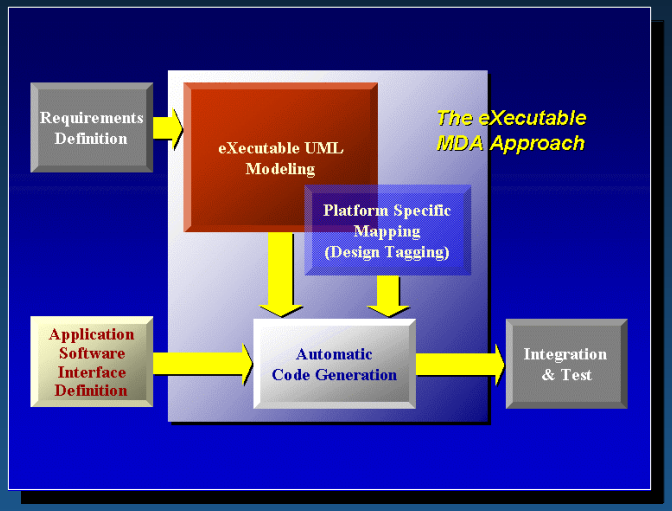
- **Nearly all implementation-specific design tasks (all but the design decisions represented by design tag values) are performed by the code generator, not the software developers**

Portable Application Software Products



The Portable Products
(and therefore the Configured Products to be placed in an Enterprise-Level Software Reuse Library)

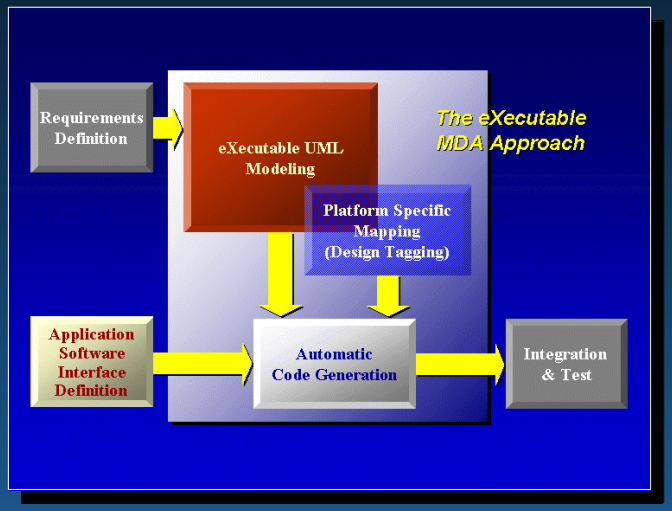
Advantages of the MDA with xUML Approach



Increased Quality

- The majority of software developers are isolated from implementation details, allowing them to focus on a thorough analysis of the application space
- Maintenance of the application source code is eliminated, while maintenance of the xUML models is ensured
- Defect injection (and the resulting rework) is reduced by automating the software phase in which most defects are injected
  - *On a typical program, after Requirements Definition approximately 2/3 of the defects are injected during implementation (coding)*

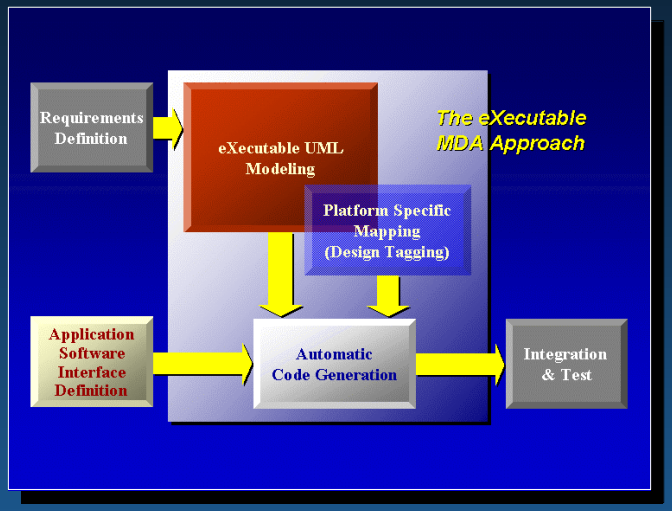
Advantages of the MDA with xUML Approach



Increased Productivity

- **Rework is reduced**
  - ***Early validation through simulation reduces rework***
    - Increase in eXecutable UML modeling span time is more than offset by decrease in Integration & Test span time
  - ***Higher quality implementation (due to automation) reduces rework***
- **Software development span time is reduced by automating the implementation phase**
  - ***Application Software development schedule is reduced by at least 20%***
  - ***The code generator, not each software developer, performs the majority of implementation-specific design tasks***
    - 40-60% of physical source code

Advantages of the MDA with xUML Approach



Cross-Platform Compatibility

- **One Application Software xUML Model database may be reused (as is) on any platform for which a mapping is defined (ie: a code generator is developed)**
  - ***xUML models are compatible with any hardware platform, any Software Execution Platform, and any Application Software Interface***
  - ***xUML models are compatible with any implementation language***

The Goal of Cross-Platform Compatibility of Application Software is Attainable with the OMG's MDA supported by eXecutable UML

Projects Using MDA with 100% Code Generation

- BAE Systems: Stingray torpedo MLU (Ada 95)
- TRW Automotive: vehicle stability system (efficient C, OSEK)
- Siemens Metering: ‘intelligent’ gas meter (C)
- Thales: Nimrod MR4 crew trainers (C++)
- GD Government Systems: ATM Switch for US Army (C)
- Royal Netherlands Navy: combat systems (C++)
- Nortel Networks: Passport Voice Gateway (C++)
- GCHQ: classified distributed application (C, objectstore)
- UK NHS: patient control of access to medical records (C++)

MDA Models: Integration and Interoperability

- **“Integrate what you have built with what you will build”**
- **Models of legacy assets, COTS products and new developments**
- **…are integrated to create platform independent models of entire systems or services.**
- **Models can exist as legacy code or a set of executable interface definitions - UML is not the only suitable formalism**
- **Interoperability is defined at the model level not the middleware level**

MDA is "Stakeholder Oriented Development"

- **MDA models are accessible representations of stakeholder knowledge**

- **MDA models formalise specialised knowledge:**
  - ***End-User knowledge about the enterprise***
  - ***Systems and IT knowledge about platforms***
  - ***Programmer knowledge about best development practice***

- **MDA models (in xUML) are working prototypes which support requirements elicitation from stakeholders**

MDA Models are Strategic Long-Life Assets

- **“A 20 year, standards-based software architecture”**
- **Wouldn’t it be nice to know that your strategic assets are correct ?**
- **They need to be tested**
- **They need to be executable**
- **UML becomes executable when used with an action language**
- **xUML models are finished when they execute their acceptance test correctly**
- **xUML models are validated specifications which offer a strong contractual basis for implementation by external contractors or in-house teams**

MDA Models Run Anywhere, Now and in the Future

- **“Tracking the next best thing”**
- **Platform Independence makes models future proof, capable of being ported to faster, better, cheaper platforms as they become available**
- **Standardised mappings for common platforms are built into MDA tools**
- **Specialised mappings to target any platform are specified with configurable code generation tools.**
- **Automated mappings guarantee interoperability between systems deployed on different platforms**

MDA with executable UML is a Lightweight Process

- **Executable modelling offers all the benefits of code-centric development without the drawbacks:**
  - ***No redundancy in the development process***
  - ***Automated traceability***
  - ***Synchronised documentation***
  - ***Early testing***
  - ***Less rework***
- **Once established, MDA shortens timescales and reduces cost**
- **MDA results in an accumulation of software assets to be integrated into future developments rather than rebuilt**

TOGAF and MDA: Outline Synthesis (& agenda for further work)

QuickTime™ and a TIFF (Uncompressed) decompressor are needed to see this picture.

use cases for business scenarios

Business Process Models (CIMS)
Domain models

Platform Independent Models (PIMS)

(Testable against business scenarios)

Platform Models

Translators