Codagen™

The evolution of development

CODAGEN TECHNOLOGIES AND MODEL-DRIVEN ARCHITECTURE (MDA)

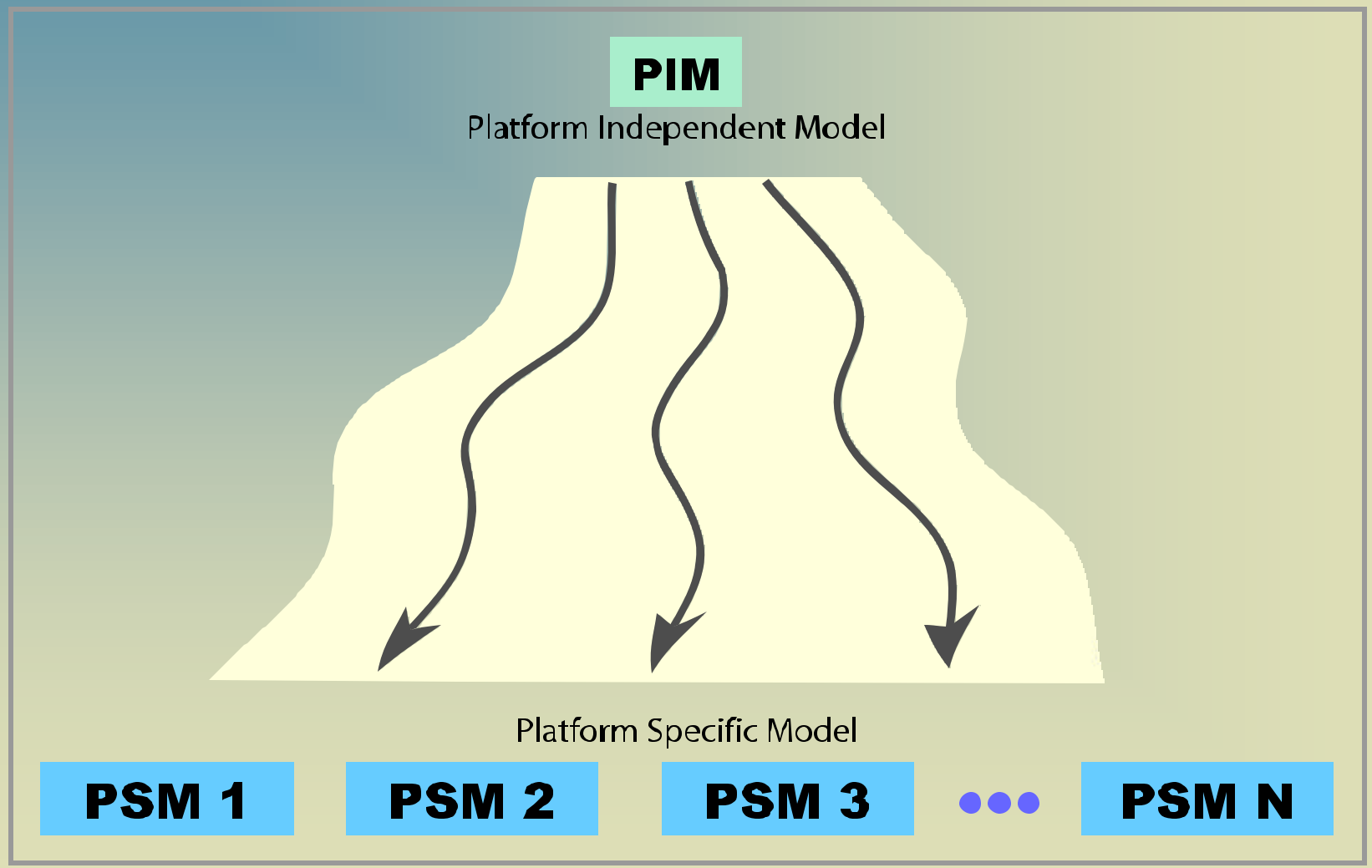
March 2002

info@codagen.com
www.codagen.com

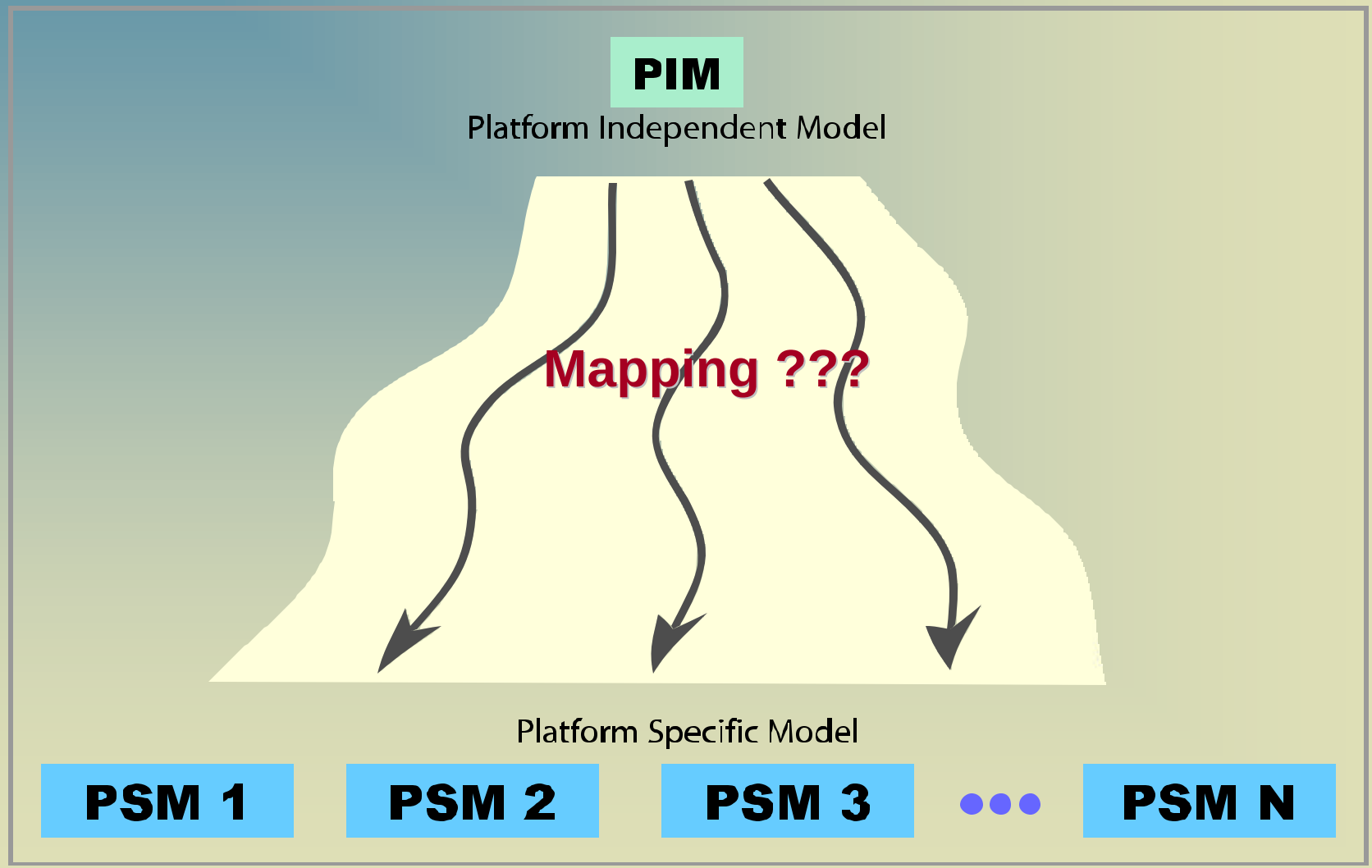
Agenda

- *OMG's MDA*
- *Gap between the PIM and code PSM*
- *Codagen's MDA Approach*
- *Benefits of the Codagen's Approach*

OMG's MDA



Gap between the PIM and code PSM



Gap between the PIM and code PSM

Platform Independent Model

UML Modeling Tools

Rational

TogetherSoft

Microsoft

Telelogic

Sybase

Embarcadero

Popkin Software

Generate < 1% code

Code Generation ???

Platform Specific Model

Integrated Development Environment

Java: *WebGain*, *IBM*, *Sun*, *Oracle*, *Borland*

VB: *Microsoft*

C#: *Microsoft*

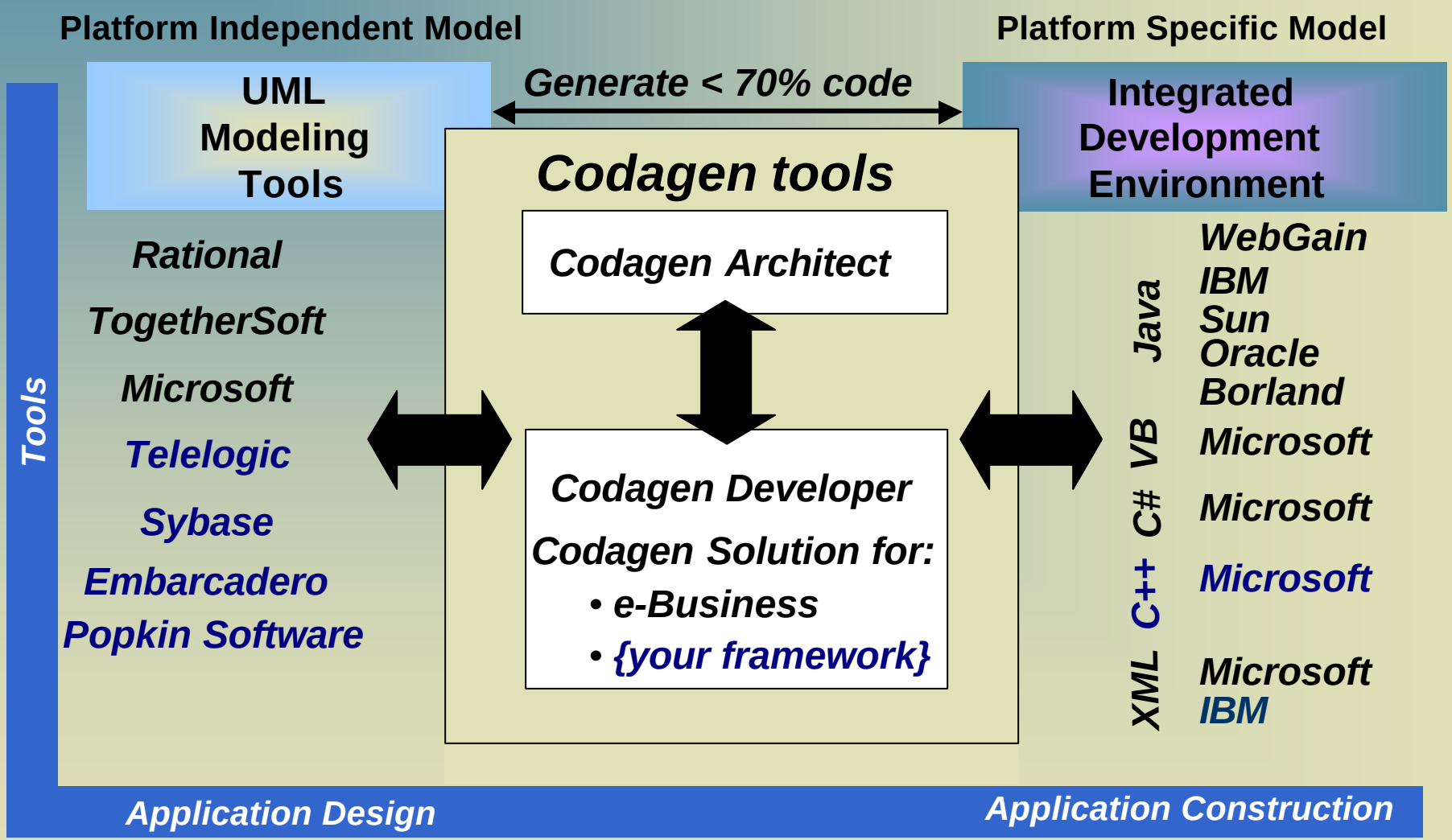
C++: *Microsoft*

XML: *Microsoft*, *IBM*

Tools

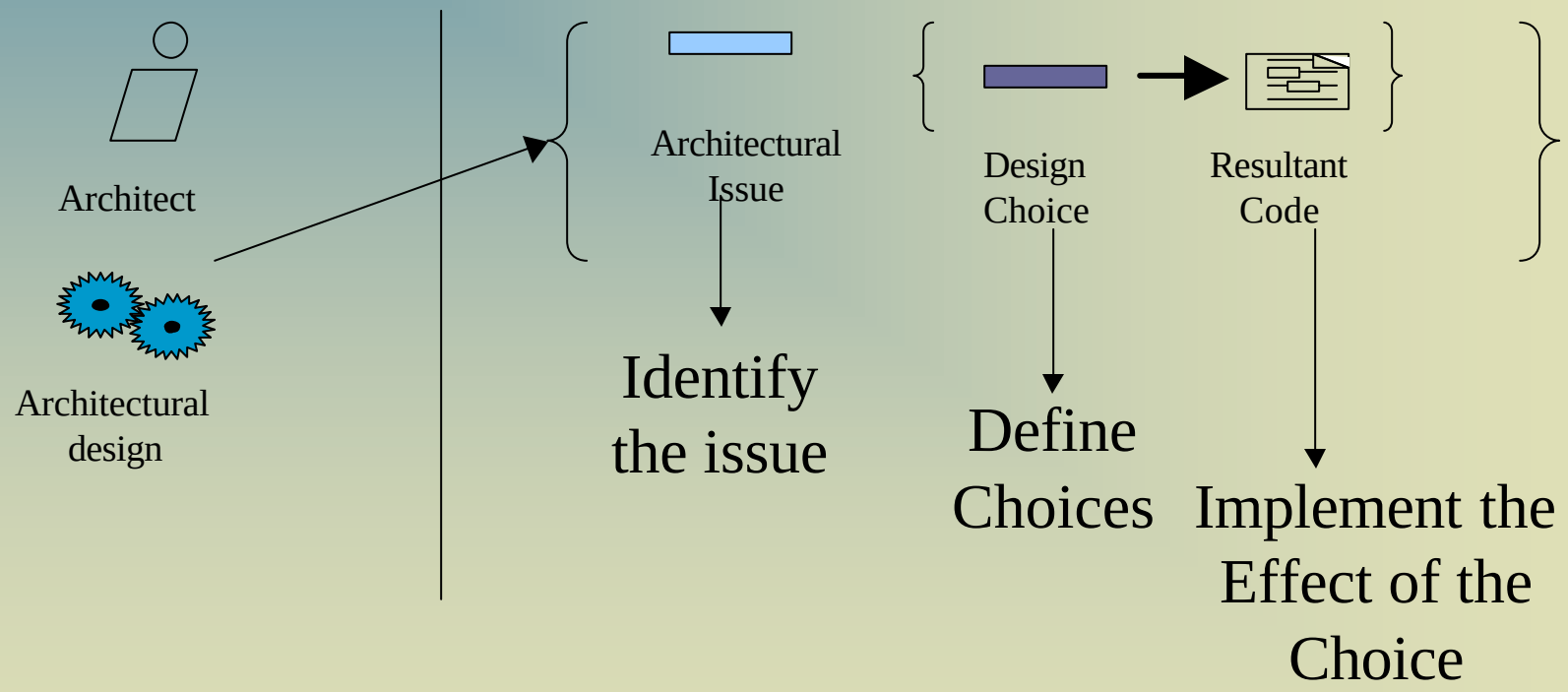
Application Design — *Application Construction*

Codagen's MDA Approach



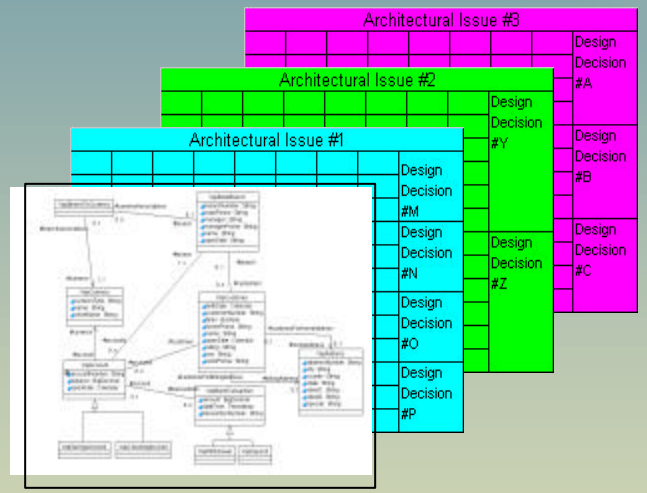
Codagen's MDA Approach

Architecture Parameterization



Codagen's MDA Approach

Application Blueprint



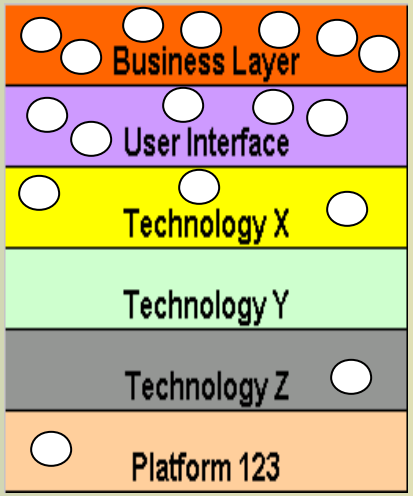
UML Platform Independent Model

Codagen's Architecture Specification

+

Codagen's Architecture Implementation

=

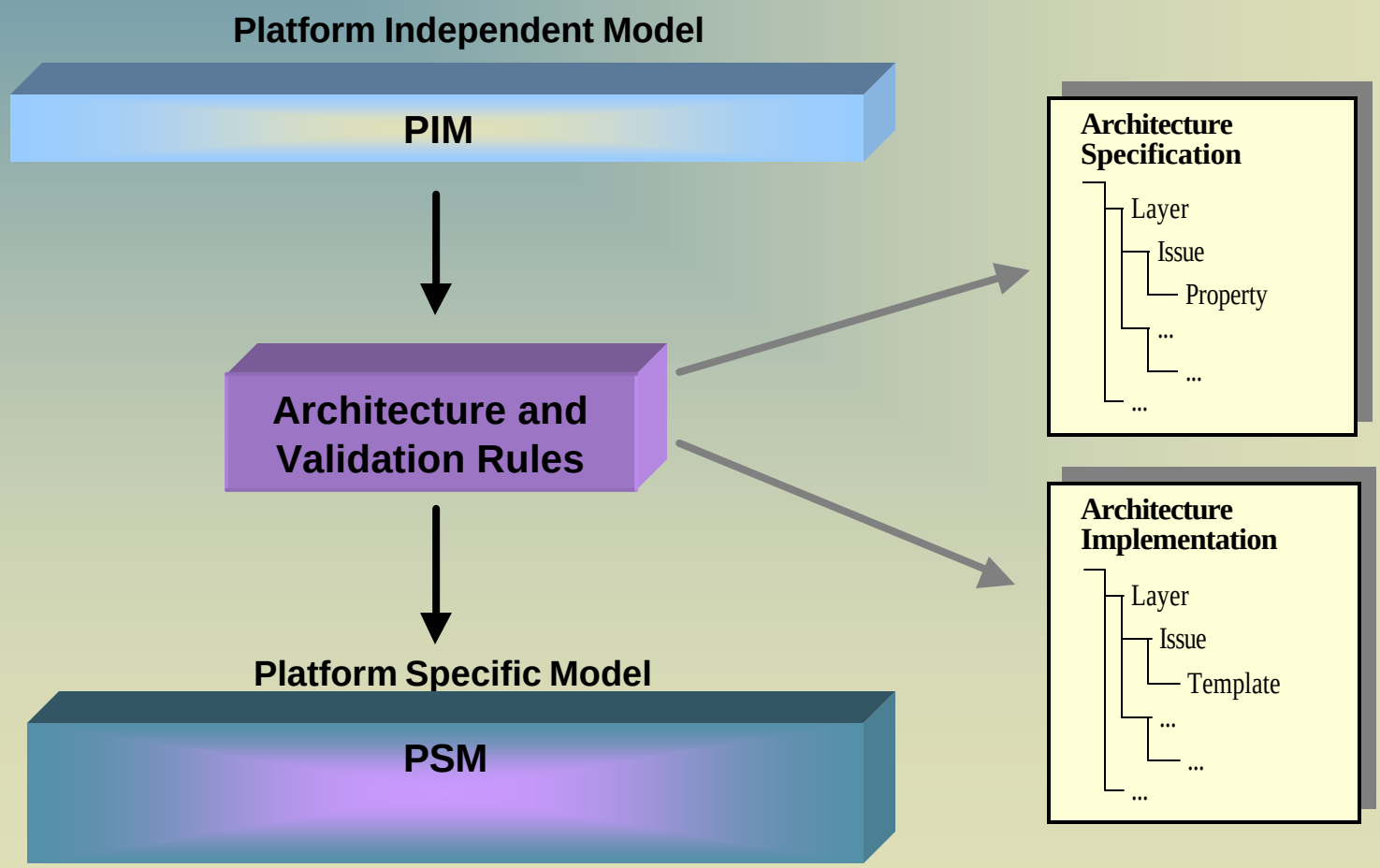


100% of the code associated with the architecture of the application

The white circle represents the value added code of the application to be manually coded by developers

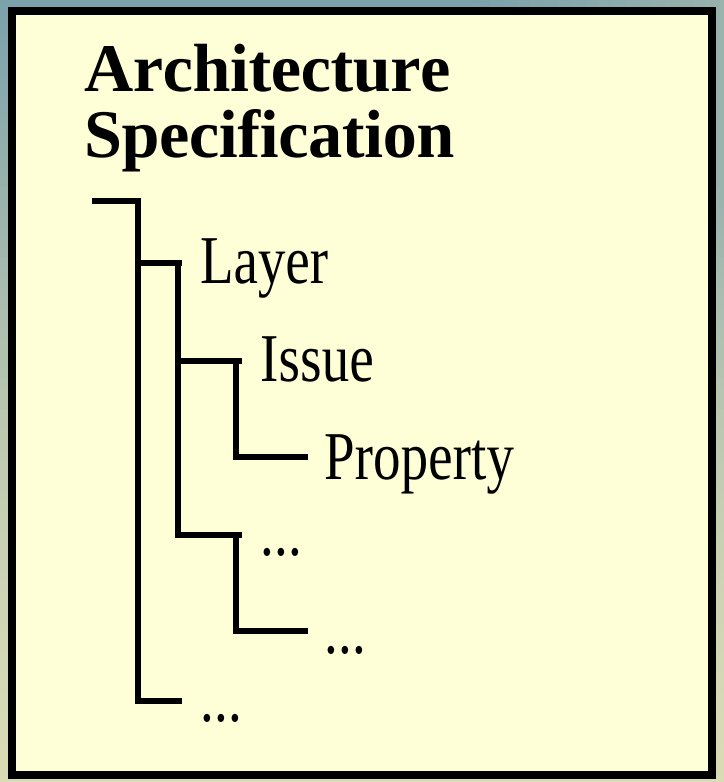
Codagen's MDA Approach

Architecture Decoupling



Codagen's MDA Approach

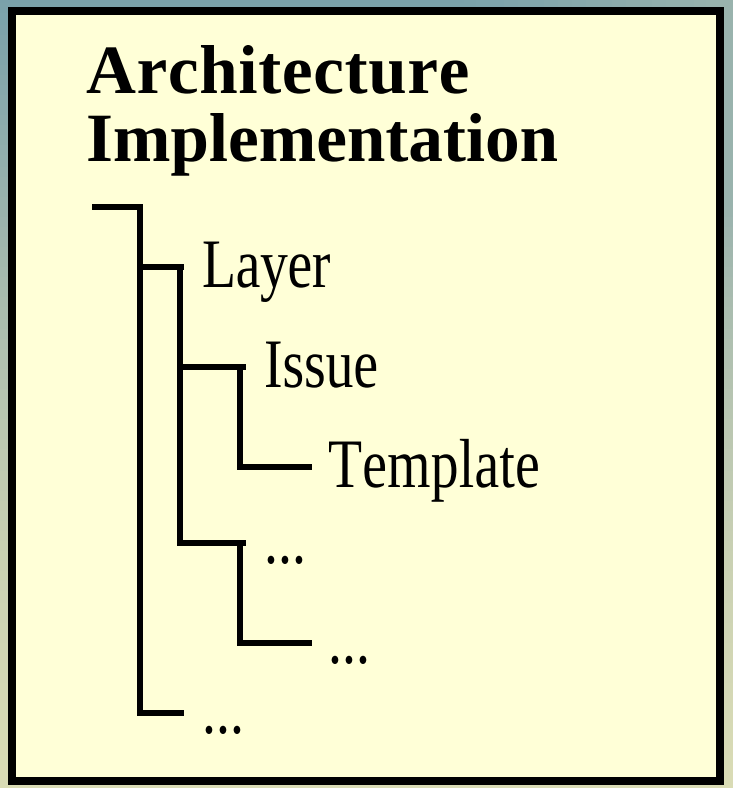
Architecture Specification



- The layers assist the architect to organize architectural issues at the appropriate level of abstraction.
- Properties are design decisions that :
  - Architect map to one or more UML elements in the PIM and define default value that can be different for each stereotype value of that UML element
  - Developer or designer needs to address

Codagen's MDA Approach

Architecture Implementation



- The architecture implementation comprises a series of declarative XML-based templates that generate a code PSM.
- Templates resolve architectural issues for a specific layer.
- Layer can be exchanged with another one.
- Code pockets provide architecture extensions.

Codagen™

The process: Design phase

- The architect specifies the architecture
  - **Layers**: Defines the different layers involved in the implementation of the business domain captured in the PIM.
  - **Issues**: For each layer, defines the architecture issues that need to be addressed in order to get the best implementation of a technology associated with that layer.
  - **Design decisions**: For each architecture issue, defines one or more design decisions (properties) that will assist the senior developer or designer to address the issue.
  - **Mapping**: Maps each design decision to one or more UML elements in the PIM (package, class, attribute, operation, and so forth).
  - **Default values**: For each value pair of design decision and UML element, defines a default value that can be different for each stereotype value of that UML element.
- Architecture specification is stored in an XML document

Codagen™

The process: Implementation phase

- The architect implements the architecture
  - **XML-based code generation templates**: Creates an implementation of the desired code PSM using the architecture specification in conjunction with Codagen's XML-based code generation templates.
  - **Rules for mapping**: Creates templates which embody the rules for mapping between the PIM and the code PSM.
  - **Key design decisions validation**: Uses the templates to capture the architecture implementation and validate the architecture specification.
  - **Source code generation**: Automates the production of the platform-specific model in source code.
  - **Interchangeable layers**: Can exchange one implementation layer with another one and generate a code PSM by layer.
  - **Architecture extension**: Provides 'code pockets' into which developers can insert value added code by hand.

Codagen™

The process: Generation phase

- The developer generates code PSM
  - **Mapping**: Maps the architecture design decision with the UML elements of the PIM
  - **Generation**: Uses the PIM and the architecture model to generate the code PSM
- The developer adds value-added code by hand in ‘code pockets’
  - **Regeneration**: Preserves value-added code when the code is regenerated because code pockets are non-disruptive for developers.

Benefits of the Codagen's Approach

Benefits for the CIO

- Provides a time-to-market solution for the the development and evolution of an application
  - Up to 70% of the code PSM is being produced with the best implementation practices over the underlined infrastructure.
- Lowers the cost of development, maintenance and enhancements
  - It makes possible to upgrade, modify or customize applications quickly and easily at any stage of the development life cycle.
- Standardizes the development process
  - It provides a better ROI on current investment made in:
    - UML and IDE tools,
    - In-house and commercial frameworks

Benefits of the Codagen's Approach

Benefits for the CIO (continued)

- Provides the agility to better react
  - Enables redeployment on different (or newer versions of) platforms or technology
- Protects organization's valuable intellectual property
  - Architects use the white box tool to capture, enforce and manage the architecture specification and implementation as well as standards and procedures.
  - Developers use the black box tool to validate the parameterized architecture and generate the code associated with the architecture.

Benefits of the Codagen's Approach

Benefits for the Architects

- Provide an opened and non-disruptive approach for code generation
  - Architects can define, view, evolve, enforce and document an architecture specification and implementation.
- Furnishes a loosely coupled architecture
  - Business requirements and technology can evolve at their own pace.
- Produces 'application solution' templates
  - Application architecture can be reused across many projects.
- Models business domain layer and generate all of them
  - This pragmatic approach streamlines the development process. Domain layer model (PIM) is not encumbered by implementation details.

Benefits of the Codagen's Approach

Benefits for the Developers

- Generates code as good as the best developer
  - Best developer is cloned programmatically.
- Reduces the skills needed for a developer
  - Architects can define common infrastructure services that a developer who is not a master of the underlying technologies can reuse.
  - Developers implement the unique end-user use cases modeled in the PIM.