



Model Driven Software Development

Craig Caminos

Outline

- The model driven theory
- Example situations
- Case studies and demonstration



Challenges

- Flexibility
- Consistency
- Productivity

The Software Equation

**Software
Architecture**

+

Design

=

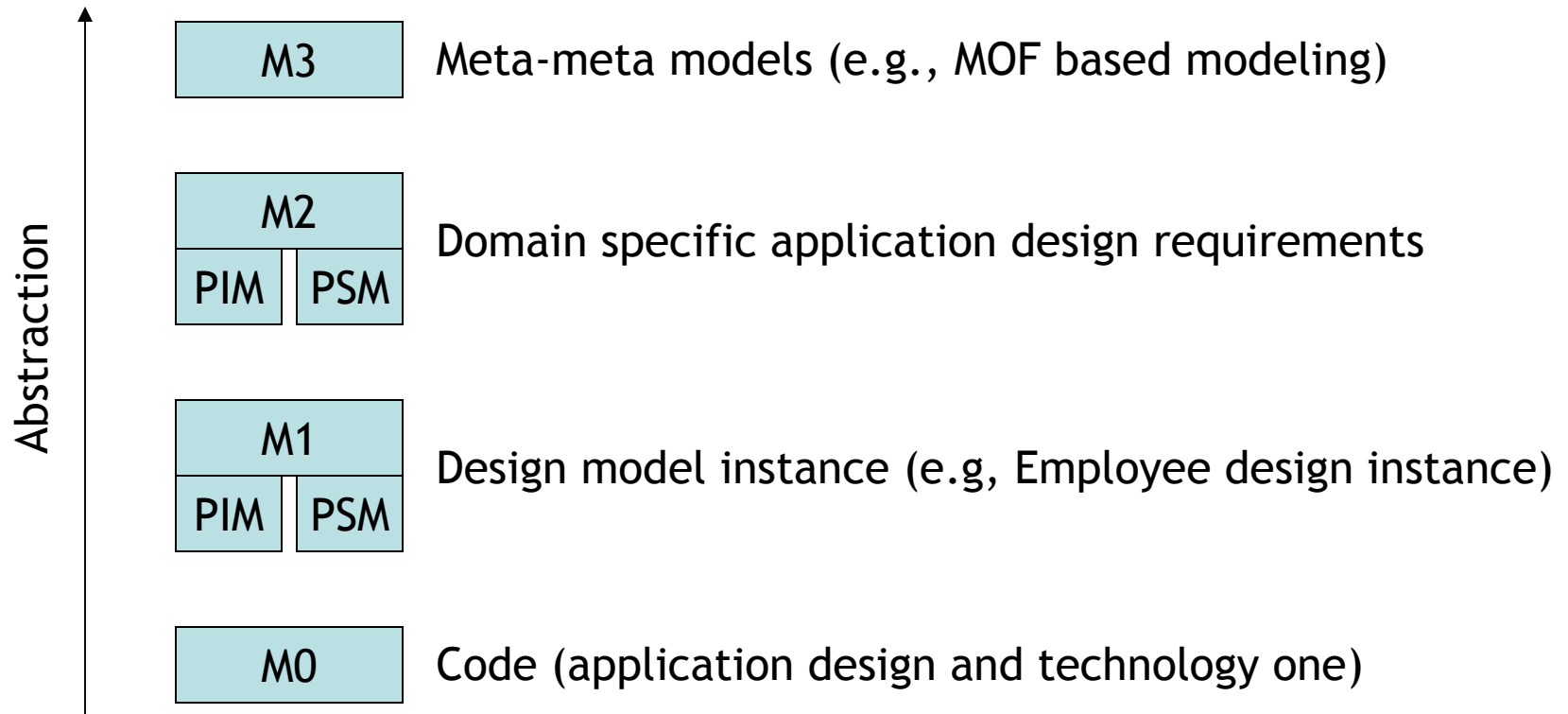
Implementation

Concepts

- Modeling
- Separate application logic from underlying technology framework
- Abstraction
 - PIM
 - PSM
- Configurable template based generation

Abstract Modeling: Overview

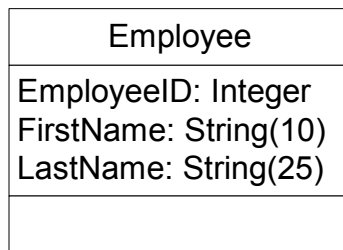
“Model driving” is produces more value given abstraction.



Low level models are not suitable for “model driving”.

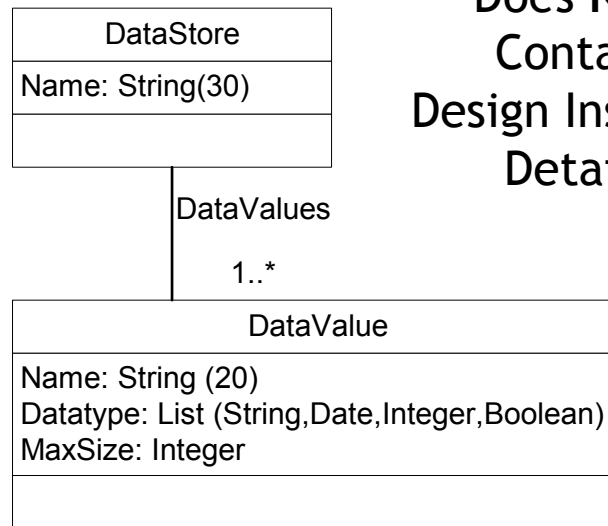
Design Abstraction

M1:



**Contains
Design Instance
Details**

M2:



**Does Not
Contain
Design Instance
Details**



Whiteboard

Relevant Situations

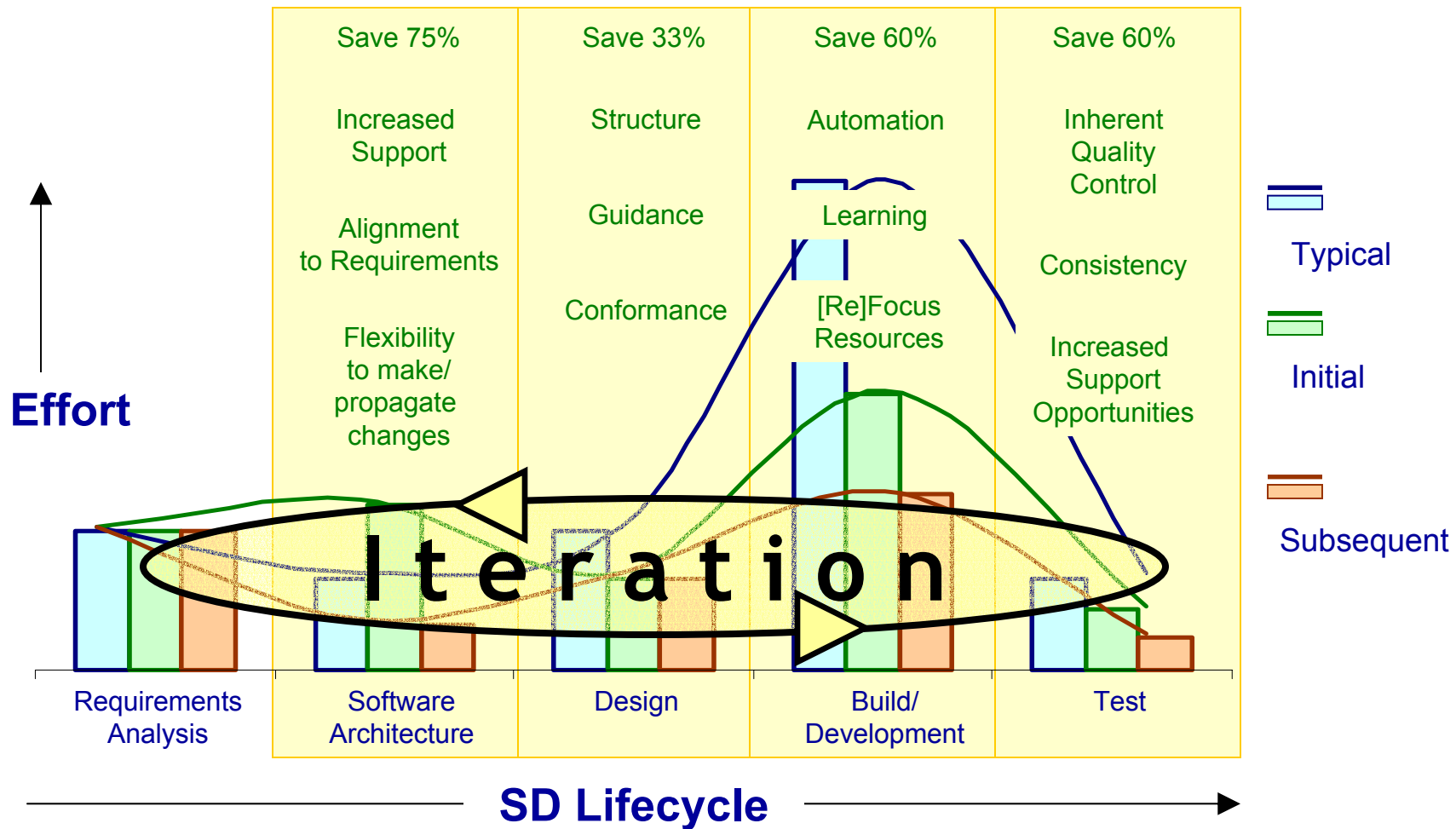
- Development
- Integration
- Migration
- Prototyping
- Technology Research
- Securing Intellectual Assets

Quick Demo: e-GEN



- A general purpose Model Driven tool from Gentastic! (www.gentastic.com)
- Demo: Web Application Development

Benefits



Case Studies



- E-government initiative - Wellington
- API suite/architecture - Denver/Auckland
- Other examples
 - Custom workflow solutions - Brisbane
 - Web architecture migration - Auckland
 - Integration architecture for NZSE - Auckland
 - Electronic procurement exchange - Auckland
 - DB administration - Auckland
 - Mobile solutions development - Auckland
 - Web application development - Auckland

E-government Initiative



When and Where: 2000, Wellington

Objective: Architect and develop an integration architecture to link several government agencies. Use a 3-tier Java and EJB architecture, using J2EE, BEA WebLogic Server 5.1, TopLink for Java and Oracle8i. Use BEA TUXEDO, Jolt and JMS to integrate various government agency systems.

Challenge: Place emphasis on business design and development. Allocate key skills on business value-add areas of development rather than production of the integration architecture and code. Use an abstract domain model to structure capture of integration design requirements, thereby streamlining the process, and generate code for processing e-documents through the integration architecture.

Model Driven Impact: Required roughly 25% of the resources otherwise needed for developing the integration architecture and code for processing e-documents. Supplemented approach with generation to generate ~95% of code required to meet the integration requirements. Associated testing effort dropped by ~50% due to higher confidence in code (i.e., noticeable consistency across output).

API Suite/Architecture



When and Where: 2002, Denver/Auckland

Objective: Package vendor sells a new system into a large US conglomerate. US conglomerate wants to integrate it with dozens of other “legacy” systems but the new package fails to offer any coherent integration architecture or suite of APIs to enable this. Commit to providing an XML/J2EE compliant integration architecture and 50 APIs in 6 months.

Challenge: With limited resources having XML/Java experience and a project plan pushing the resource requirements past 30 people and 12 months, the team needs to increase productivity.

Model Driven Impact: Able to run in *parallel* the architecture design, the API design process, and development, compressing the effort. As architecture was being defined, analysts designed API requirements using XML schemas - the structure of which mapped to abstract domain model. The XML schemas later applied to PSM once ready. Using code generation, total effort took only 6 months and 15 people.

Additional information



www.omg.org/mda

For MDA specifications (an example)



www.gentastic.com

For e-GEN: a general purpose solution

CAMINOS

craig.camino@camino.co.nz

021 707 178

For additional information and expertise