



| Application Services, IBM Austria

Model Driven Architecture

Necessary prerequisites and consequences

Michael Pichler, IT Architect, IBM Austria
mpichler@at.ibm.com

June 20, 2006

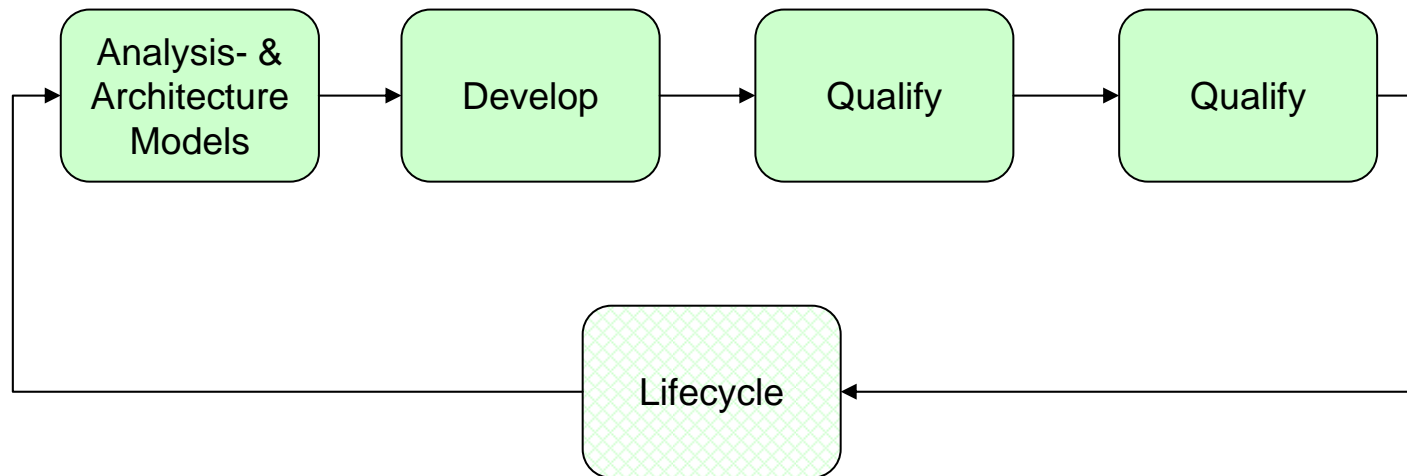
© 2006 IBM Corporation

Agenda

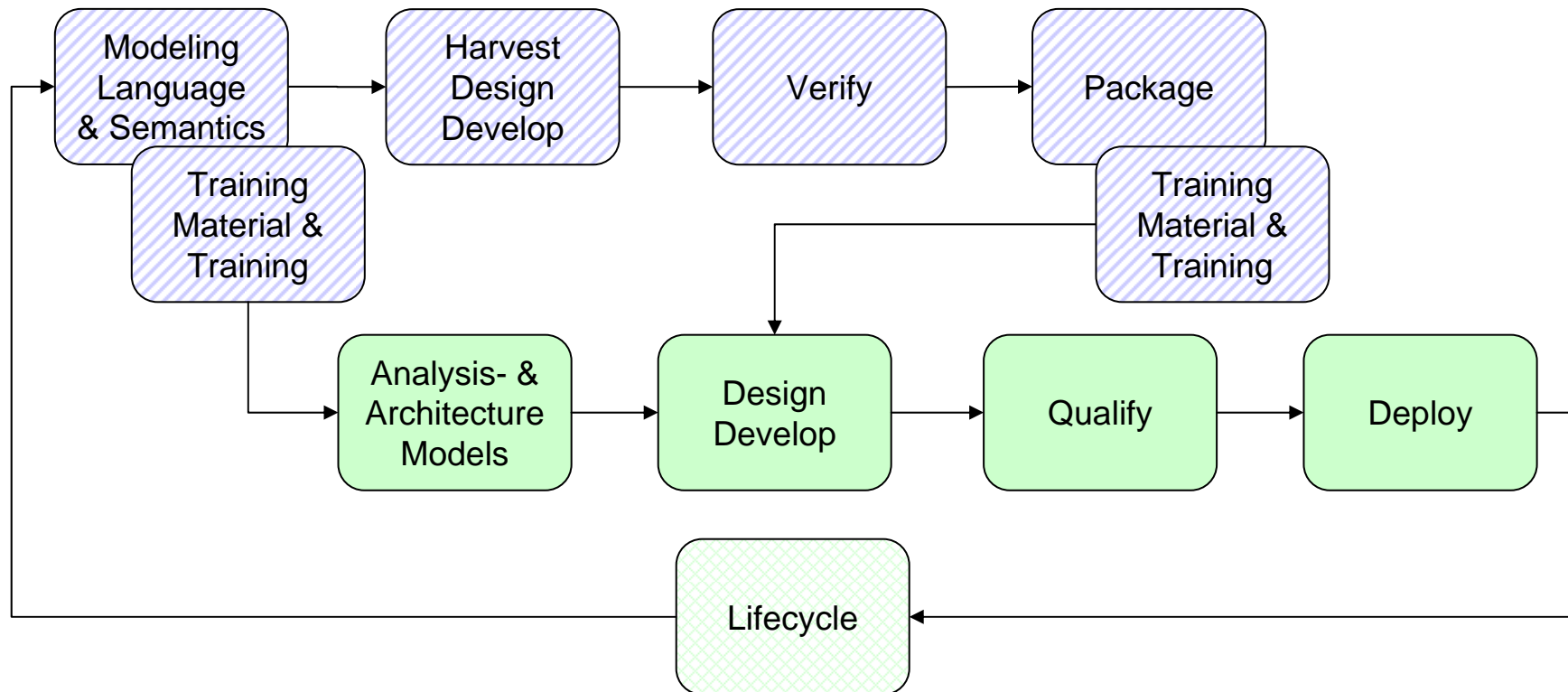
- Organizational Aspects
- Games with Numbers – Project simulations

Organizational Aspects

Software Life-Cycle

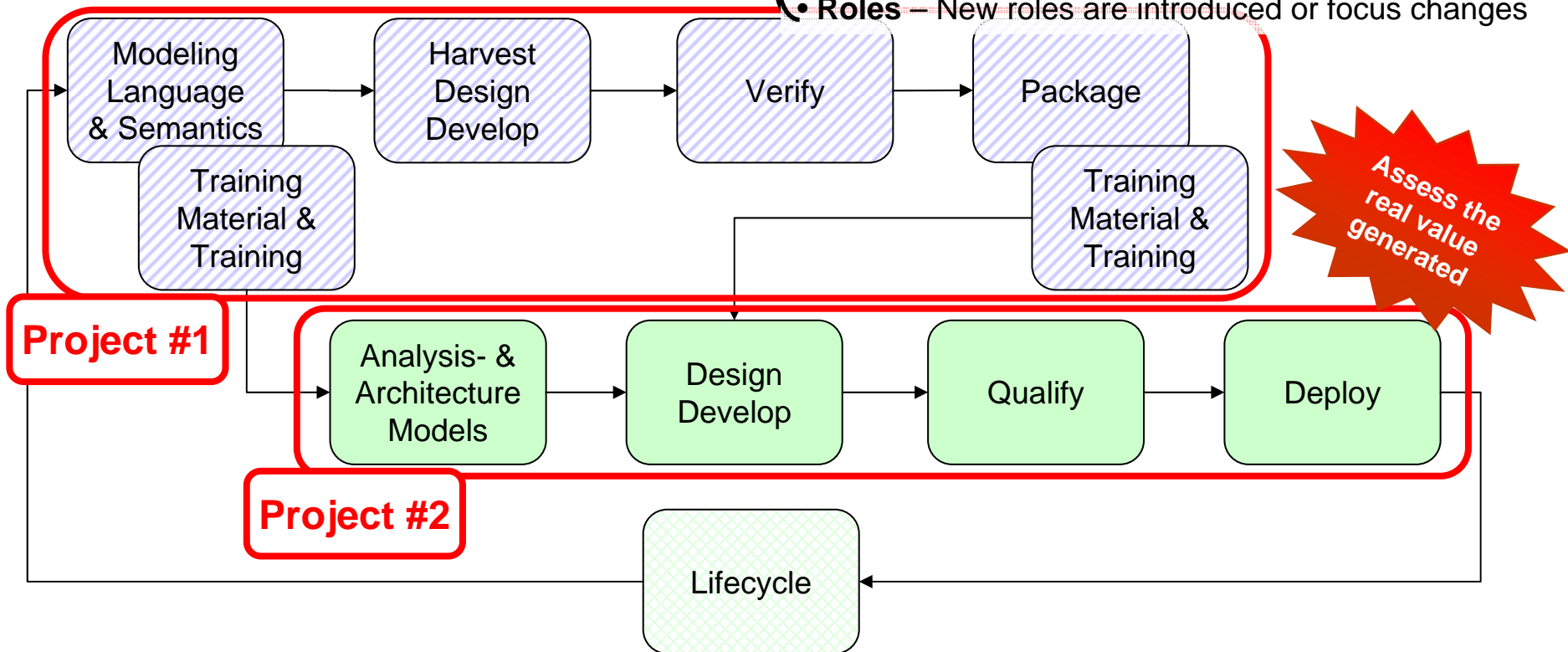


Software Life-Cycle with MDA



Management and Planning Aspects

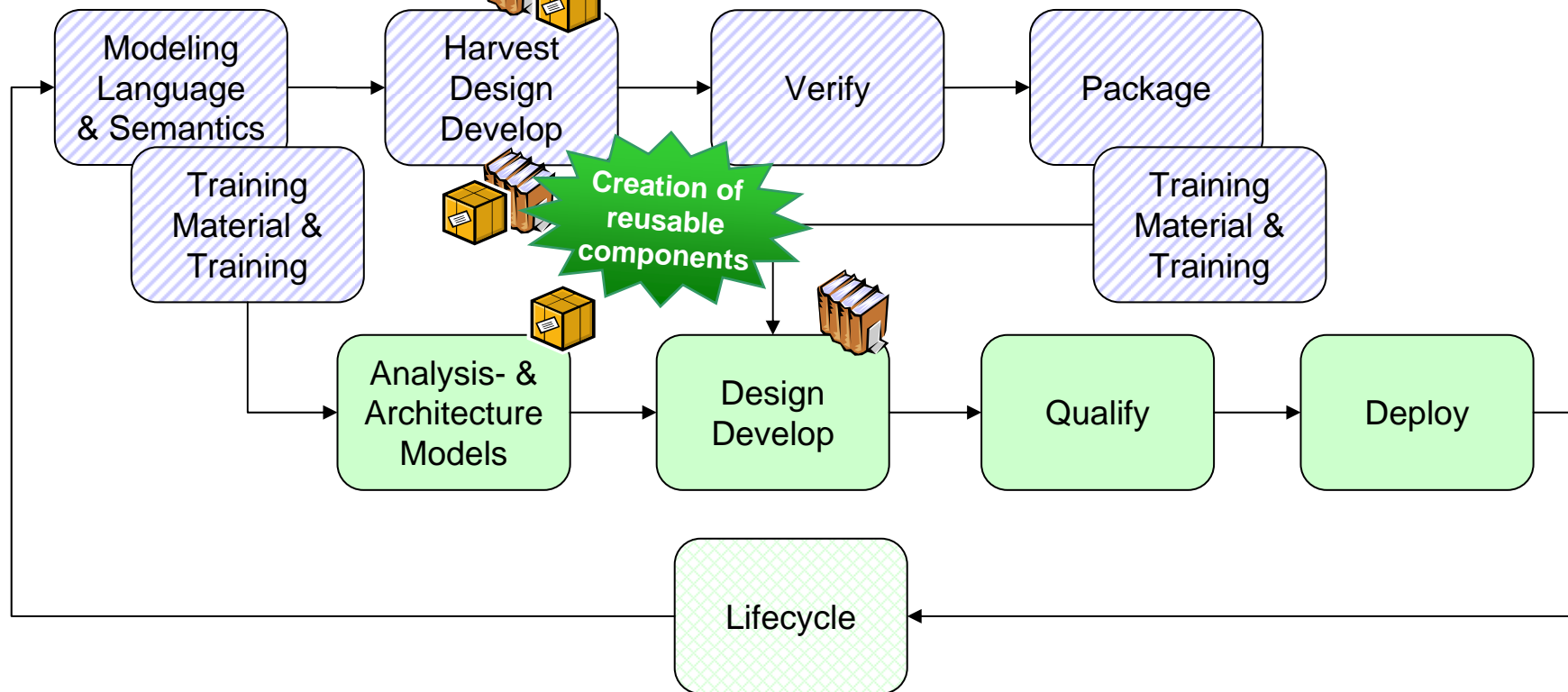
- **Challenge** – Both projects must be planned and executed as SW development projects
- **Schedule** – Make sure MDD tooling projects is inter-locked with business application project
- **Setup** – Whether or not to combine MDD with business application development
- **Skills** – Plan for building MDD skill for project team (Business Analyst, Solution Architects, MDD Tooling Developer, Business Application Developer)
- **Tracking** – Just like any other SW dev. Project
- **Roles** – New roles are introduced or focus changes



Reuse Aspects

Reuse of assets, fragments, patterns and transformations

- **Actively seek for reuse** - Reuse must be fostered by the organization by introducing mandatory tasks into the process
- **Searchable** – Assets are published in a searchable way
- **Asset future** – Shared assets have clear ownership and funding
- **Standards** ! – Common standards must be in place or developed to enable reuse across projects



Games with Numbers – ROI?

Basis for Simulations

Projects simulated

- **Assessed Software Development project** – Abt. 8 person years in size
- **MDD Tooling project** – About 8 person years in size as well

Project settings

- First **I**, multiple (subsequent) 8 PY software solutions are developed by a Software Services organization using MDA and then handed over for life-cycle
- Second **II**, a software product or a long-running software solution (multiple 8 PY release cycles) is developed using MDA

To be shown → Employing MDA is a gain – What do you think?

Important Notes:

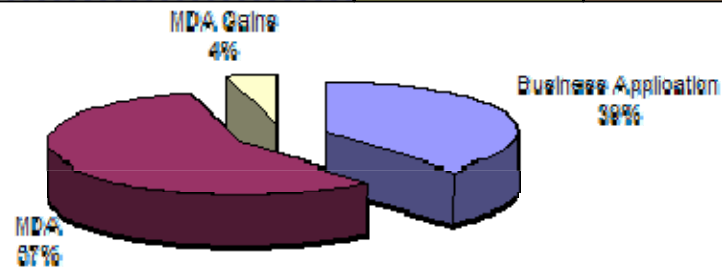
- Simulations are a simplistic view of the world
- Peripheral efforts/costs not considered (e.g. Licensing, etc.)



Project #1 – Additional MDA Efforts

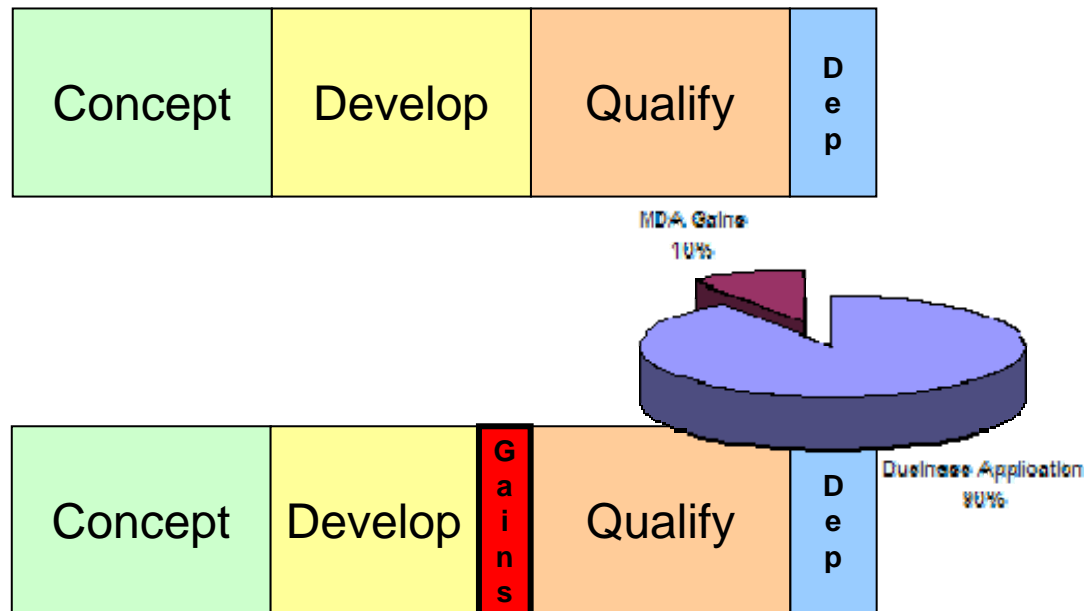


- Development of model language and Semantics (business & technical)
- Modeling training
- Design of transforms
- Development of transforms
(*Model to Project, Model to Deployment Script, Model to Build Script, Design Model to Business Objects, UI Interaction Model to UI Stub or UI and others*)
- Testing of Modeling and Transforms





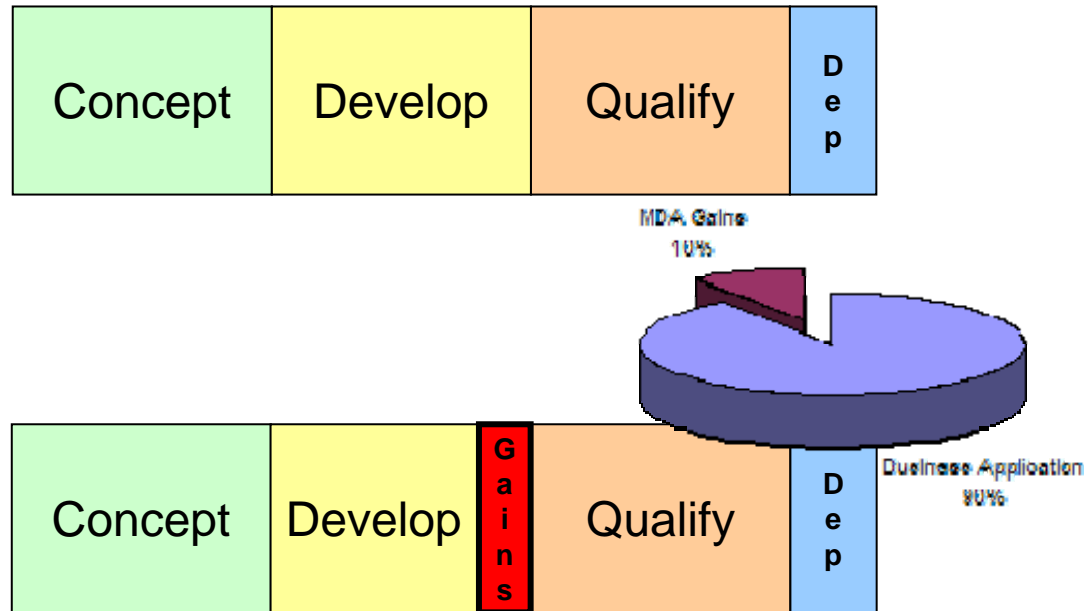
Project #1 – Minimal gains through MDA



- No gains in Concept Phase
- Modeling focused on particular areas – *Business logic is not modeled*
- Gains in Develop Phase – *Due to Model to Code transformation*
- No or marginal gains in Qualify – *Qualify focus is on testing of business logic*
- No or marginal gains in Deployment
- Significant investment into building MDA framework



Project #2, #3, #4, etc. – The unexpected



- No gains in Concept Phase
- Modeling focused on particular areas – *Business logic is not modeled*
- Gains in Develop Phase – *Due to Model to Code transformation*
- No or marginal gains in Qualify – *Qualify focus is on testing of business logic*
- No or marginal gains in Deployment
- Significant investment into building MDA framework

Assumptions:

- No adaptations of the MDA framework are necessary
- Domain Analysis, Domain Design, Architecture is not reusable



Lessons learned – Why only marginal gains?

- Software **Services projects** are often of **highly individual character**
- Based on the assumptions no **ROI** until after about **20 projects or 18 years**

Factors for increasing gains

- **Repeatable projects** – If developed solutions – or parts of them – can be reused in follow on projects higher gains may be possible
- **Standards** – Standards within an organization will make MDA assets reusable across projects and make gains possible
- **Solution life-time** – If developed solutions live long enough gains can be located through reuse of existing model-, and code-base

Important Note: MDA gains in terms of productivity and quality have been kept low



Lessons learned – Why only marginal gains?

- Software **Services projects** are often of **highly individual character**
- Based on the assumptions no **ROI** until after about **20 projects or 18 years**

Factors for increasing gains

- **Repeatable projects** – If developed solutions – or parts of them – can be reused in follow on projects higher gains may be possible
- **Standards** – Standards within an organization will make MDA assets reusable across projects and make gains possible

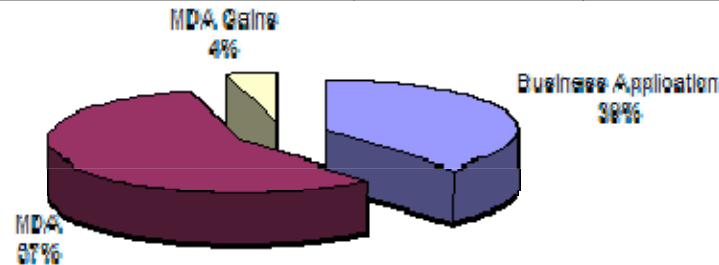
Important Note: MDA gains in terms of productivity and quality have been kept low



Project #1 – Additional MDA Efforts

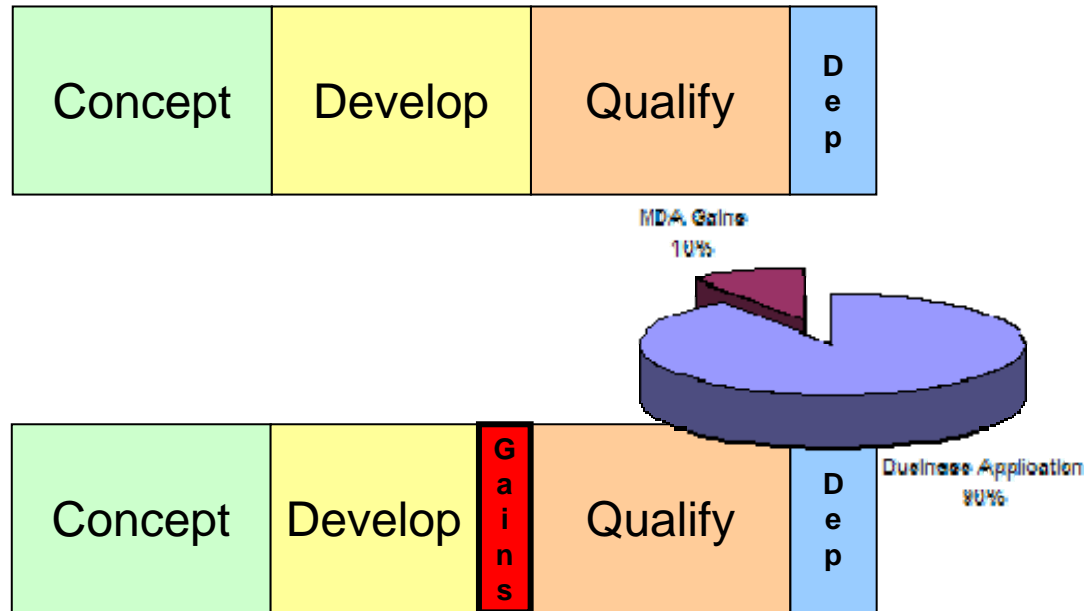


- Development of model language and Semantics (business & technical)
- Modeling training
- Design of transforms
- Development of transforms
(Model to Project, Model to Deployment Script, Model to Build Script, Design Model to Business Objects, UI Interaction Model to UI Stub or UI and others)
- Testing of Modeling and Transforms





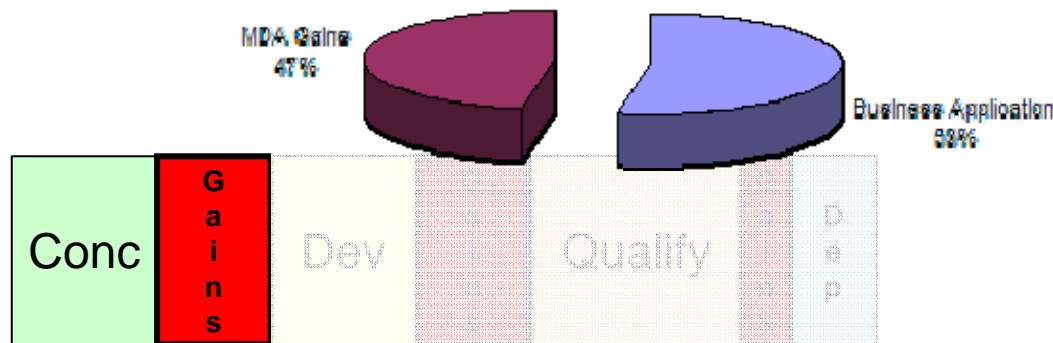
Project #1 – Minimal gains through MDA



- No gains in Concept Phase
- Modeling focused on particular areas – *Business logic is not modeled*
- Gains in Develop Phase – *Due to Model to Code transformation*
- No or marginal gains in Qualify – *Qualify focus is on testing of business logic*
- No or marginal gains in Deployment
- Significant investment into building MDA framework



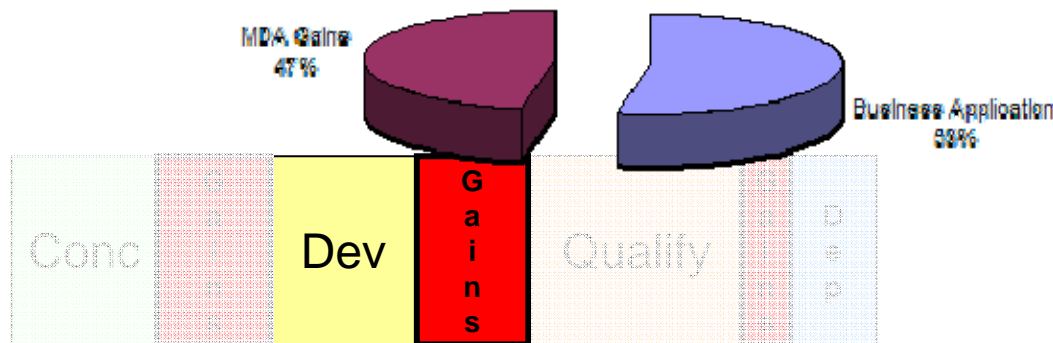
Project #2 – Significant gains through MDA



- Concept Phase – 70% of Models can be reused
- Develop Phase – About 60% of code can be reused
- Qualify Phase – 30% of Test efforts can be saved through already developed & tested code
- No or marginal gains in Deployment



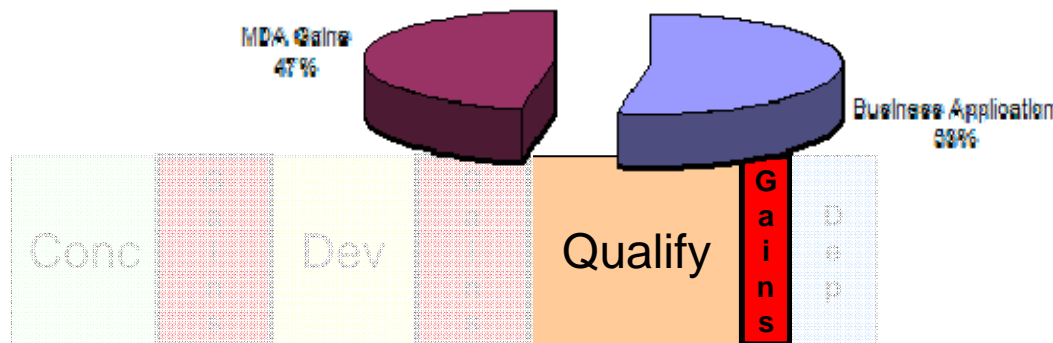
Project #2 – Significant gains through MDA



- * Concept Phase – 70% of Models can be reused
- Develop Phase – About 60% of code can be reused
- * Quality Phase – 30% of Test efforts can be saved through already developed & tested code
- * No or marginal gains in Deployment



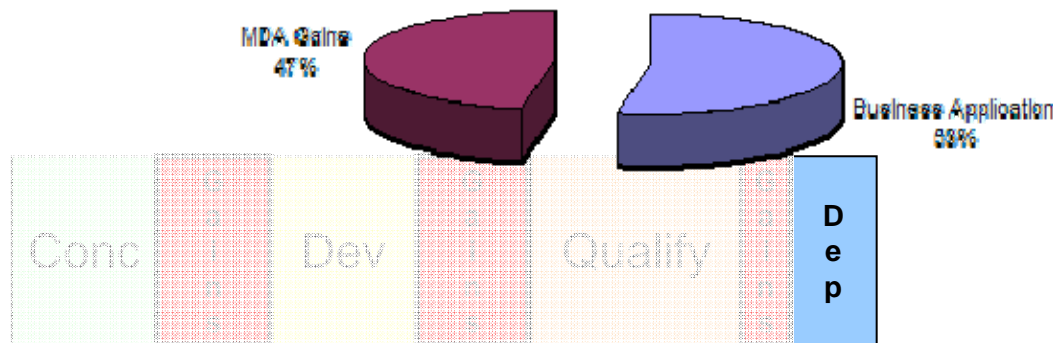
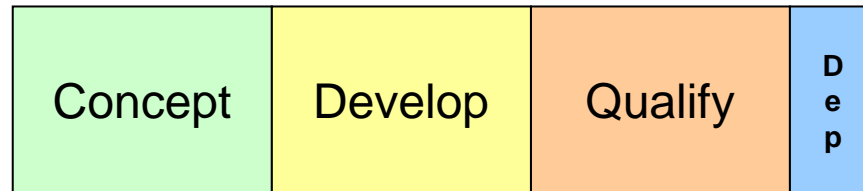
Project #2 – Significant gains through MDA



- Concept Phase – 70% of Models can be reused
- Develop Phase – About 60% of code can be reused
- Qualify Phase – 30% of Test efforts can be saved through already developed & tested code
- No or marginal gains in Deployment



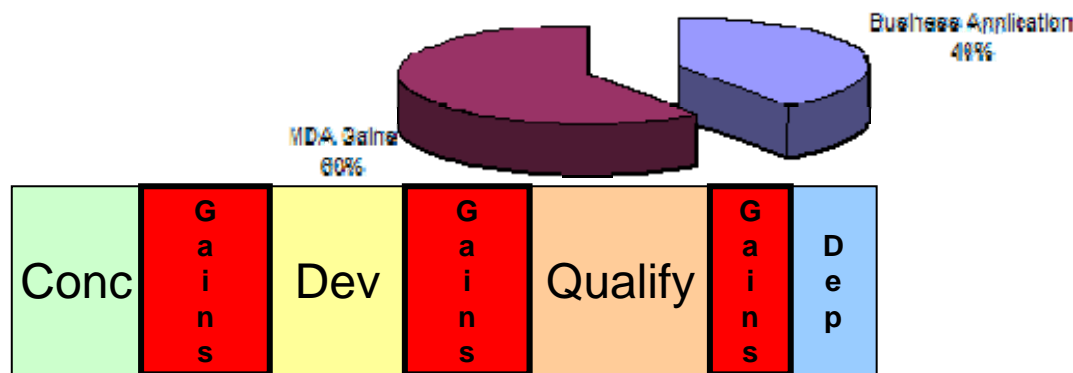
Project #2 – Significant gains through MDA



- * Concept Phase – 70% of Models can be reused
- * Develop Phase – About 60% of code can be reused
- * Qualify Phase – 30% of Test efforts can be saved through already developed & tested code
- No or marginal gains in Deployment



Project #3 – More gains through MDA



- Concept Phase – 80% (vs. 70%) of Models can be reused
- Develop Phase – About 70% (vs. 60%) of code can be reused
- Qualify Phase – 50% (vs. 30%) of Test efforts can be saved through already developed & tested code
- No or marginal gains in Deployment



Lessons learned – Why significant gains?

- **Product Development projects** or **Long-Living Software Services projects** are much more suitable for MDA as reuse can be fostered
- Based on the assumptions **ROI** after about **3 projects or 3 ½ years**

Factors for increased gains

- **Assets** – *Model fragments and transforms created during project can be reused in later stages*
- **Quality** – Gains through – over time – increased quality of generated code
- **Solution life-time** – The solution life-time of the product or solution is long enough

Important Notes:

- Business logic is still not modeled and therefore also not generated.
- MDA gains in terms of productivity and quality have been estimated high

Summary

- MDA is most suitable for **Product Development** and **Long-Living Software Solution** projects
 - Assets can be reused
 - Quality gains through – over time – increased quality of generated code
- **Reuse** of MDA Frameworks, MDA Assets or MDA Fragments is **key and challenging** – just like defining and identifying in Asset-based development approaches
- **Development of Standards** is **essential**
- **Key to developing ‘real’ gains** is **modeling and generation of business logic**, as most of the defects are located within business logic

Thank you!