DELTA software technology
The Generator Company

www.D-S-T-G.com

The Generator Company

OOP 2006
Scalable Software Systems and Solutions
18.01.2006

Generative Solutions for the Modernization of your COBOL Applications

Delta Software Technology is a leader in advanced software generators for the integration and modernization of COBOL applications.

Architecture-Driven Modernization
Goals – Concepts – Benefits

Rüdiger Schilling
Delta Software Technology

DELTA software technology
The Generator Company

Credits

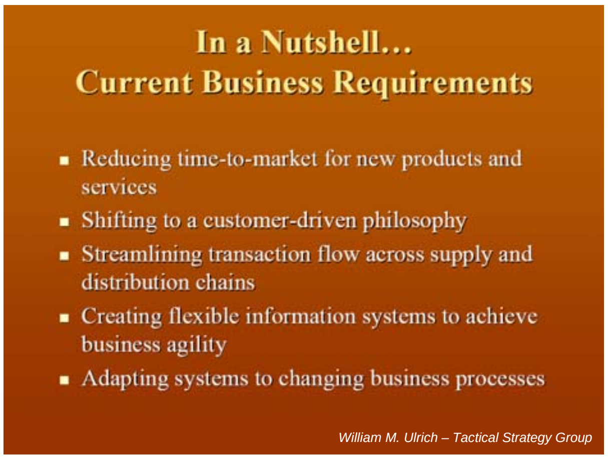
- This lecture is partially based on the tutorials and workshops of the ADM Task Force
- Special thanks go to the following colleagues:
  - William M. Ulrich, Tactical Strategy Group
  - Nicolaus Mansurov, KlocWork
  - Philip Newcomb, The Software Revolution

Architecture-Driven Modernization

- ➢ Modernization?
- Scenarios
- ADM Task Force
- Standards
- ADM Now?

Challenges

Many roads lead to ...?



In a Nutshell…
Current Business Requirements

- Reducing time-to-market for new products and services
- Shifting to a customer-driven philosophy
- Streamlining transaction flow across supply and distribution chains
- Creating flexible information systems to achieve business agility
- Adapting systems to changing business processes

William M. Ulrich – Tactical Strategy Group

Challenges

- We constantly have to face new business requirements and rapidly-changing business processes
- New technologies, architectural concepts and development paradigms arise continuously
- When we try to meet these demands, two questions come up:
  1. Which of the new technologies and concepts are the most suitable ones?
  2. What to do with the existing application; can it be further developed or are they just an obstacle we have to get rid of?
- Legacy – valuable inheritance or sins of the past?

Legacy As A Valuable Inheritance

CIO Insight – 12/2002

- What are the most important reasons why you still keep hold of your enterprise‘s legacy systems?

They still **support business processes**	54.4%
They are still **reliable**	49.7%
We have **manpower available** for the support	44.3%
They are still **more cost-effective** than the alternatives	41.6%
No budget available for any changes	36.9%
They still **support** our **strategic objectives**	36.9%

On the one hand ...
The legacy systems are still relevant and reliable

DELTA software technology
The Generator Company

Legacy As Sins Of The Past

CIO Insight – 12/2002

- If you are currently migrating your legacy system, or intend to do so, what are the most important reasons?

New strategic **objectives**	65.2%
The legacy system **does not support** the business processes **sufficiently**	59.9%
The legacy system **does not support** the current **strategic objectives**	56.1%
New systems are **more cost-effective**	48.5%
Legacy systems **provide limited interoperability**	41.7%

... and on the other hand
Migration and transformation are being planned

DELTA software technology
The Generator Company

Investments: Concrete Example CICS

- Investments in number of applications
  - Globally, 16,000 enterprises (amongst them 490 of the Fortune 500 Companies) use CICS.
  - There are about 30 million CICS terminals installed.
  - Performing about 20 billion transactions per day.
  - CICS transactions are processing an amount of 64 trillion dollars (10^{12}) per day.
- An estimated calculation based on the following assumptions:
  - 20,000 S/390 servers comprise an average of one million lines of active application code (between 200,000 and 50 million pro server), makes a total of 20 billion LOC.
  - Productivity of 2,000 LOC per man year, Investment of 10 million of man years.
  - 100,000 $/man year, investment in S/390 application software: one trillion $ (10^{12})
  - For comparison: the GNP of the USA 1999 comprised a total of 9 trillion $.

Prof. Wilhelm Spruth, Tübingen

COBOL Legacy - What The Analysts Say

- Gartner Group:
  - 75% of all business data is processed in COBOL.
  - There are between 180 billion and 200 billion lines of COBOL code in use worldwide. – This represents over 60 percent of the world’s computer code.
  - Existing legacy systems are predominantly written in COBOL.
  - 15% of all new applications (5 billion lines) through 2005 will be in COBOL.
  - "Integration with Legacies" is the number one concern of IT managers in 2003.
  - Over the next four years there will be a 13% decrease in their number [of COBOL developers] due to retirement and death.
- The COBOL Report:
  - CICS transaction volume (such as COBOL-based ATM transactions) grew from 20 billion per day in 1998 to 30 billion per day in 2002.
- Tactical Strategy Group:
  - Replacement costs for COBOL systems, estimated at $25 per line, are in the hundreds of billions of dollars.
- Giga Group:
  - There are over 90,000 COBOL programmers in North America in 2002.
  - The most highly paid programmers in the next ten years are going to be COBOL programmers who know the Internet.

Contradictions

- On the one hand:
  - According to a survey by CIO Insight 48.5% of the CIOs consulted assume that new systems are more cost-effective
- On the other hand, Informatik Spektrum, April 2003 reported:
  - The effort for the maintenance and continuous adaptation of legacy systems to the ever-changing company demands is significant, but there is one amazing observation: **maintenance costs for COBOL programs are significantly lower** than with other programs. The Y2K costs per Function Point are estimated at an average of **45$** for all languages; whereas for COBOL programs it is only **28$**. This is one reason why COBOL is often also the first choice for new applications. The existing number of COBOL programs (about **180 billion lines of code) thus grows by a further 5 billion lines of code each year**.
- What does this mean?
  - Will new systems once again be built with COBOL?

Quality Problems

- The crash of Arianne 5 was due to a register overflow
- The breakdown during the introduction of the London Ambulance system
- The breakdown of the master computer during the re-opening of Hamburg railway station (1995)
- “Hartz-IV” Software (2004)
- Deutsche Bahn (German Railways) – The total breakdown of the reservation system caused by a software update (2005)

How Can We Cope With The Challenges?

- The usual suspects:
- 1st approach: Develop everything new
  - With a new architecture, new technologies, new employees
  - We are now exclusively using …
- 2nd approach: Enterprise Application Integration
  - Broker, Hub and Spoke
  - Middleware is the silver bullet
- 3rd approach: Software packages
  - “Standard” software instead of in-house development
  - We are now exclusively using ... SAP

DELTA software technology
The Generator Company

1st Approach: Develop Everything New

- Costs and risks are almost always underestimated
- 85% of IT projects are finished too late or not at all*
- Only 9% of IT projects are on schedule and within budget*
- ERP projects: the implementation lasts for years,
  35% are aborted and rarely completely adopted*

  * Standish Group International
- Each year, 16.5 billion US $ are spent on systems that will never be put into production (Information Week)
- **Both the market and users call for new functions – never for new software**

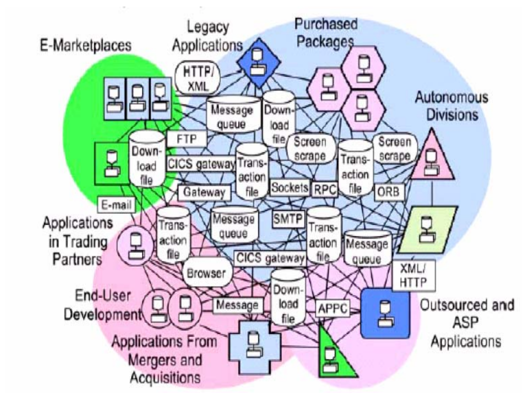
DELTA software technology
The Generator Company

We Are Now Exclusively Using ...

- New technologies (language, middleware etc.)?
  - OMG EAI Workshop 2001:
    All efforts to replace N technologies by 1 new technology usually end up with N plus 1 technologies

    The greatest advantage of new technologies is usually that their disadvantages are not yet known
- The SOA paradox
  - Objective: faster fulfilment of business demands
  - How to get there: First bring the complete development to a standstill to introduce a new technology, and then reimplement the existing applications once again

2nd Approach: Enterprise Application Integration?



Gartner Forecast Newsletter, 2001

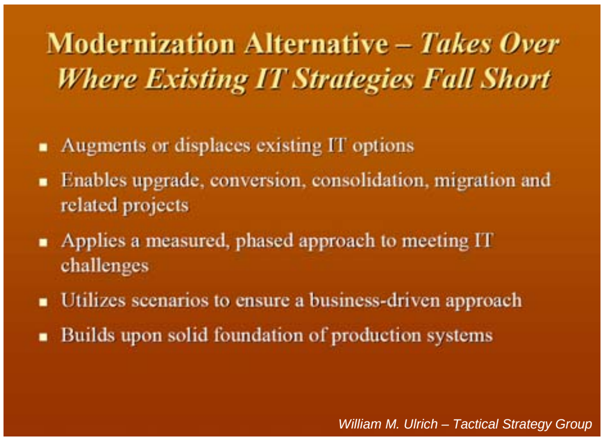
- Vendor-specific integration solutions result in an increasing number of new legacy layers
- Apparently: EAI architectures increase the complexity of application systems ...
- ... and may in the end lead to chaos

All These Approaches Are Lacking In Foresight

- Complete replacement of the applications:
  - Much too expensive, too time-consuming and too hazardous
- Enterprise Application Integration
  - Does not solve the problem of the actual architectural restrictions
- Standard software* – not the magic bullet:
  - 35% of projects are terminated early
  - Characteristically only 40% of the promised functionality is provided
  - Only 10% of projects are realized on schedule and within budget

  ** Server World Magazine, June 2001*
- Outsourcing: moves the problems but does not solve them

The Alternative


Modernization Alternative – *Takes Over Where Existing IT Strategies Fall Short*

- Augments or displaces existing IT options
- Enables upgrade, conversion, consolidation, migration and related projects
- Applies a measured, phased approach to meeting IT challenges
- Utilizes scenarios to ensure a business-driven approach
- Builds upon solid foundation of production systems

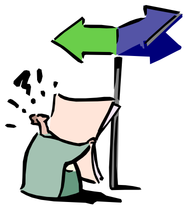
William M. Ulrich – Tactical Strategy Group

Since the usual approaches are not sufficient,

the alternative of "modernization" is the only sensible approach.

What does "modernization" actually mean?

Architecture-Driven Modernization



☑ Modernization?

➢ Scenarios

ADM Task Force

Standards

ADM Now?

Modernization: Three Basic Activities

- Evaluation
  - Analysis and publication of the system- and business artefacts architectures, data and process flows, system structures and behaviours
- Stabilization and standardization
  - All tasks that structure, rationalize, realign, modularize or reconfigure the existing systems in any way
- Transformations
  - Extraction of data definitions, data and business rules together with the reuse of the existing system artefacts for the creation of new target architectures

http://adm.omg.org/

Scenario I
Application Portfolio Management

- Objective
  - Understanding and documenting existing applications
- Situation
  - None or only poor documentation of the application systems and their architecture
  - Detailed information is mandatory for fulfilling audit requirements and government regulations
    (e.g. Sarbanes-Oxley, Basel II)
  - IT systems will undergo major changes or are to be outsourced
  - Modernization projects require baseline information for concepts and strategy

http://adm.omg.org/

DELTA software technology
The Generator Company

Scenario II
Application Improvement

- Objective
  - To increase the robustness, integrity, quality, consistency and/or performance of applications
- Situation
  - A growing upgrade request backlog cannot be met due to system quality.
  - The system is experiencing high failure rates or reliability problems.
  - Malcontent users due to long development cycles
  - System upgrade costs are not proportionate to business returns.
  - As a preparation for an outsourcing project or for its re-integration
  - Only selective modifications e.g. field length (Y2K, account number etc.)

DELTA software technology
The Generator Company

Scenario III
Language-to-Language Conversion

- Objective
  - Conversion from one language into another
  - May also concern language versions and “dialects”
- Situation
  - A language is regarded as obsolete, is no longer vendor supported or understood by available programming talent
  - The old language does no longer meet today’s business requirements
  - A system is to be moved to a new platform which does not run the existing language or particular language version
  - A baseline system must be established from which current applications may be migrated to a strategic architecture

Scenario IV
Platform Migration

- Objective
  - Modification and adaptation of applications so they can be used on new hardware or software platforms
- Situation
  - Hardware and/or operating systems are no longer supported or viable
  - “Right Sizing“ a system by moving it to a distributed environment or back to a mainframe
  - The current platform does not support the accepted operating system
  - The platform has to be changed to gain one common platform for several systems

Scenario V
Non-Invasive Application Integration

- Objective
  - The main task is the ability of equipping legacy applications with modern front-ends.*

  * *This definition only applies to user interfaces, it has to be emphasized though, that there is smooth crossover e.g. to the Service Enablement (SOA Transformation), which might be realized non-invasive as well, or at least minimal-invasively.*
- Situation
  - Business users want to replace aging front-ends with Web-based front-ends
  - Functionality, data structures and interfaces of the core system are to remain essentially unchanged
  - An integration project is seen as a first step to subsequent modernization objectives such as a SOA transformation

Scenario VI
Services-Oriented Architecture Transformation

- Objective
  - Restructuring existing applications
- Situations
  - Business functions embedded in monolithic batch or online applications are to be provided as services
  - Complex user interfaces and data accesses complicate the isolation of business functions
  - Existing front-ends rely on segmented, inconsistent and redundant back-end systems or batch updates

Scenario VII
Data Architecture Migration

- Objective
  - Migration of non-relational file and database systems to relational data architectures
- Situation
  - Users are increasingly experiencing data inconsistency, integrity problems and less data accessibility
  - Existing ISAM, hierarchical and network structures are not suitable for distributed modern systems
  - Users require more flexible views of business data
  - Older file or database systems are obsolete and will no longer be supported

Scenario VIII
Architecture Consolidation

- Objective
  - Redundant applications are to be consolidated and integrated
- Situation
  - After a merger or acquisition redundant applications exist that are to be consolidated
  - Redundant business areas are to be consolidated to a single unit
  - Several systems contain large segments of overlapping functions
  - Inadequate or nonexistent sharing of data between systems reduces availability and leads to inconsistent results

Scenario IX
Data Warehouse Deployment

- Objective
  - Data-Warehouses contain data for extraction and analysis; are based on a common repository – no update
- Situation
  - Business functions require consolidated access to certain data (e.g., customer information)
  - Users require access to data that crosses organizational and application boundaries
  - Related data is defined across multiple systems, making it difficult to access and evaluate it
  - There is not enough time or budget for a “correct” integration and consolidation of the data systems

Scenario X
Application Package Selection and Deployment

- Objective
  - Facilitating analysis, selection and deployment of application packages
- Situation
  - The possibilities of the introduction of an application package are to investigated
  - An application package has already been acquired and needs to be adapted
  - A more detailed documentation of the package is required
  - It is to be investigated how the package can be interfaced or integrated with an existing system

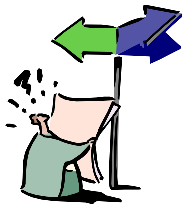
Scenario XI
Reusable Software Assets

- Objective
  - Identifying and editing reusable application functions
- Situation
  - An organization has understood the benefits and value of reuse and component-based development
  - There is a significant installed base of application systems that contain valuable functionalities that are to be turned into reusable assets or components.

Scenario XII
Model-Driven Architecture Transformation

- Objective
  - Transforming a non-model-driven application system into the direction of a model-driven architecture (MDA)
- Situation
  - An organization has decided to develop and maintain its applications in a model-based way
  - MDA concepts and tools are to be introduced or already used
  - The current data and application architecture is obsolete
  - Business users require functions that can hardly be integrated into the existing architecture

Architecture-Driven Modernization



☑ Modernization?

☑ Scenarios

➢ ADM Task Force

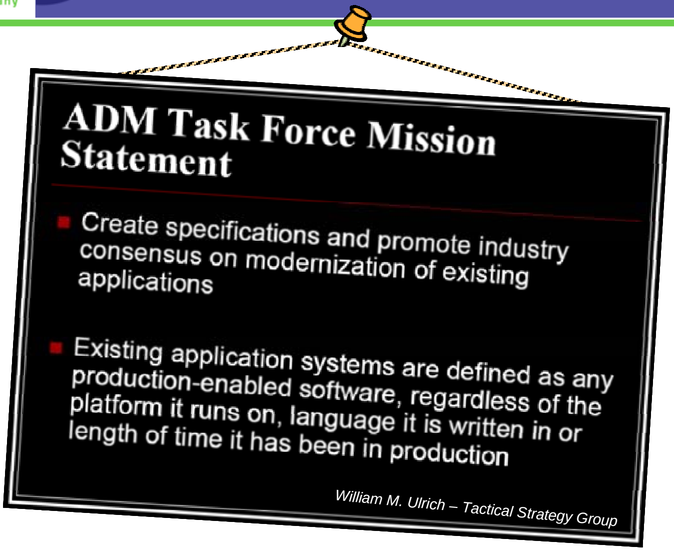
Standards

ADM Now?

Wikipedia: Abbreviation ADM Stands For …

- ... **Atomic Demolition Means**, nuclear weapons with minor explosive force
- ... **Algorithmically Discrete Mathematics**
- ... **Arbeits[illegible] Deutscher Markt** - German market research group (and social res
- ... **Archi**
- ... **Arbei** m of German
- ... A **ner**, see also ADM mass
- ... **Atmospheric Dynamics Mission**, an ESA climate sat ee also ADM Aeolus
- ... Amtrack 3-letter code for the **Ardmore** (Oklahoma) railway station

Architecture Driven Modernization
→
New OMG Standards
expanding MDA procedures and standards
to existing systems



ADM Task Force Mission Statement

- Create specifications and promote industry consensus on modernization of existing applications
- Existing application systems are defined as any production-enabled software, regardless of the platform it runs on, language it is written in or length of time it has been in production

William M. Ulrich – Tactical Strategy Group

OMG Standards

- OMG Task Force “Architecture-Driven Modernization”
  - ADMTF, founded in 2003
- The first two standards:
  - Knowledge Discovery Meta-model (KDM)
  - Abstract Syntax Tree Meta-model (ASTM)
- The Roadmap establishes seven steps for developing the standards
- Important commercial companies are participating:
  - IBM, EDS, Fujitsu, HP, Unisys ...
  - Delta Software Technology

ADM Roadmap (1)

- 1. Knowledge Discovery Meta-Model Package (KDM)
  - Establishes an initial meta-model
  - Allows tools to exchange application meta-data for the modernization – across applications, languages and platforms
  - Provides a comprehensive view of application structure and data but does not represent software below the procedural level (see ASTM)
- 2. Abstract Syntax Tree Meta-Model Package (ASTM)
  - Extends software representation below the procedural level
  - Allows the complete representation of applications
  - Facilitates the exchange of granular meta-data for conversion at the language level
  - Consolidates all syntactic language models in one common meta-model

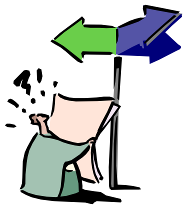
ADM Roadmap (2)

- 3. Analysis Package (AP) – (Starting 2005)
  - Facilitates the examination of structural meta-data (KDM, ASTM) to gain detailed meta-data about behaviour and structure
  - E.g. design patterns, business rules or other aspects that are not apparently components of a system but are to be semantically derived
- 4. Metrics Package (MP) – (Starting 2006)
  - Deriving metrics that describe the technical, functional and architectural attributes
  - For supporting planning, effort estimation, ROI analysis and risk assessment

ADM Roadmap (3)

- 5. Visualization Package (VP) – (Starting 2007)
  - For the (graphical) visualization of different aspects of ADM models
  - E.g. graphics, diagrams or metrics tables
- 6. Refactoring Package – (Starting 2008)
  - Describes ways of using ADM models for the (tool-supported) refactoring of application systems
- 7. Target Mapping & Transformation (TMT) Package – (2009)
  - Describes the mapping between the ADM models and the target models e.g. UML
  - From ADM to MDA

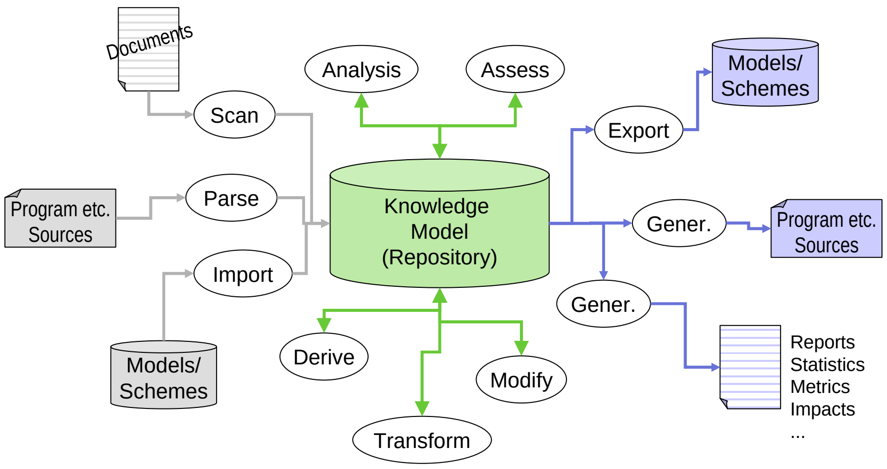
Architecture-Driven Modernization



- ☑ Modernization
- ☑ Scenarios
- ☑ ADM Task Force
- ➢ Standards
- ADM Now?

Knowledge Discovery Meta-Model (KDM)

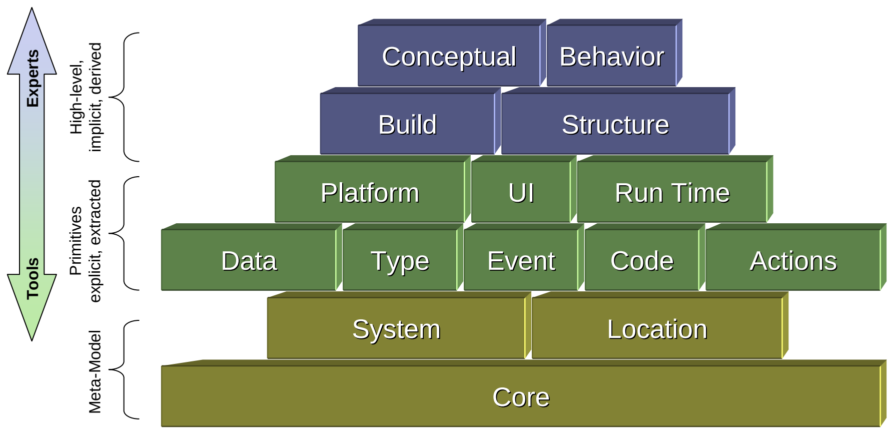
One model as the basis for all modernization activities



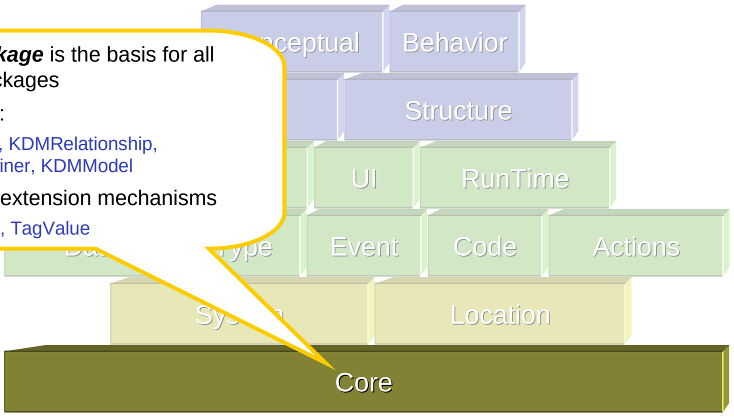
Knowledge Discovery Meta-Model (KDM)

- Each of the modernization scenarios requires, creates or modifies the data of the associated components
  - The individually required data differ
  - Basically, very different tools may access the data and in variable sequences
- This standard aims to define one common meta-model for the data to ensure the interoperability of tools of different vendors
  - The OMG specifies meta-models on the basis of MOF (Meta Object Facility)
  - MOF is the standard for defining models and meta-models as well as for the implementation of the associated interfaces

KDM Standard Building Blocks



Core Package



The ***Core Package*** is the basis for all other KDM packages

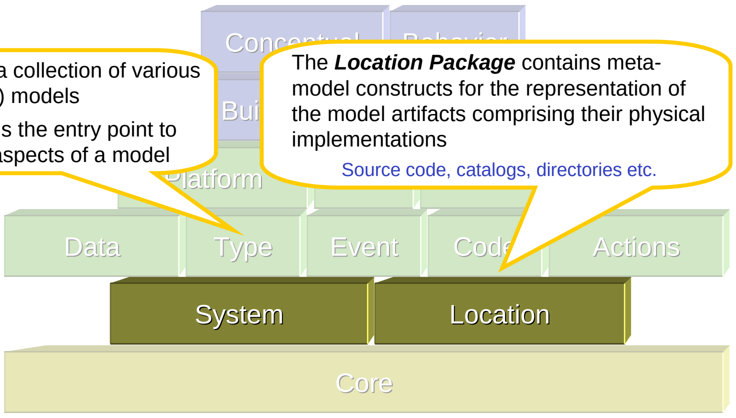
Meta concepts:

KDMEntity, KDMRelationship, KDMContainer, KDMModel

“Light-Weight“ extension mechanisms

Stereotype, TagValue

System & Location Package



A ***System*** is a collection of various KDM (partial-) models

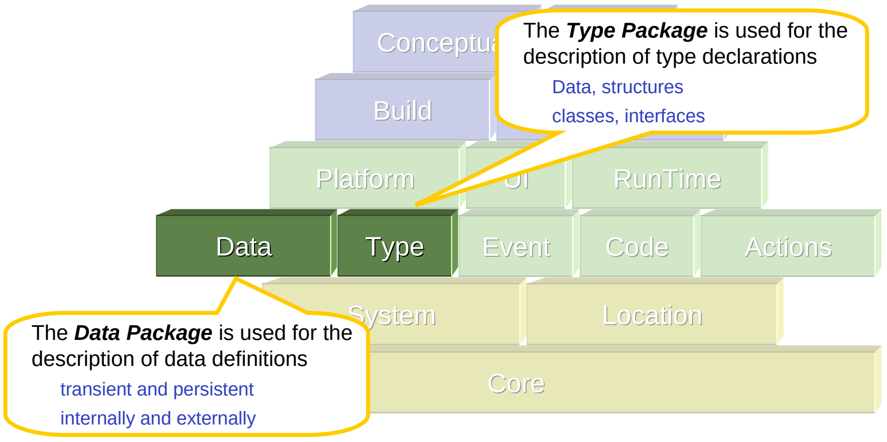
The ***System*** is the entry point to the different aspects of a model

The ***Location Package*** contains meta-model constructs for the representation of the model artifacts comprising their physical implementations

Source code, catalogs, directories etc.

DELTA software technology
The Generator Company

Data & Type Package



The ***Type Package*** is used for the description of type declarations

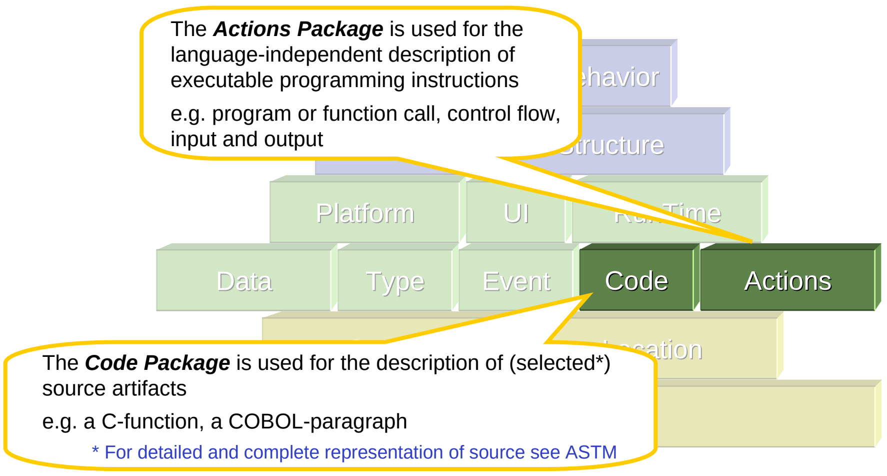
- Data, structures
- classes, interfaces

The ***Data Package*** is used for the description of data definitions

- transient and persistent
- internally and externally

DELTA software technology
The Generator Company

Code & Actions Package



The ***Actions Package*** is used for the language-independent description of executable programming instructions

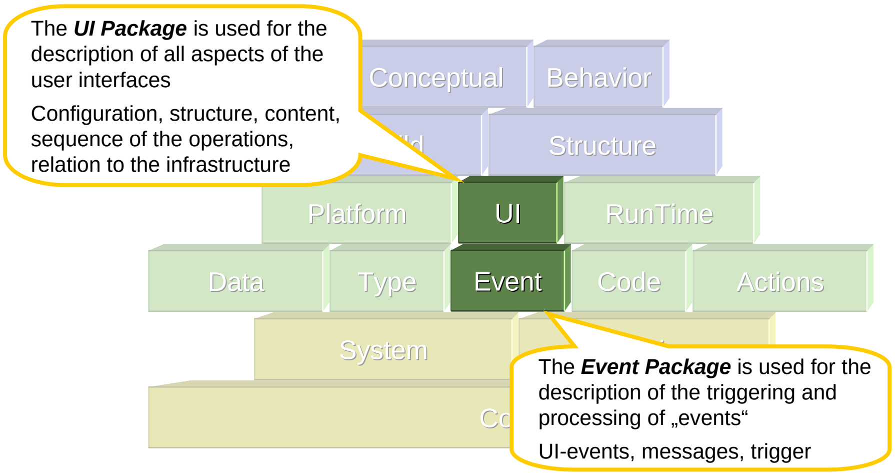
e.g. program or function call, control flow, input and output

The ***Code Package*** is used for the description of (selected*) source artifacts

e.g. a C-function, a COBOL-paragraph

* For detailed and complete representation of source see ASTM

Event & UI Package



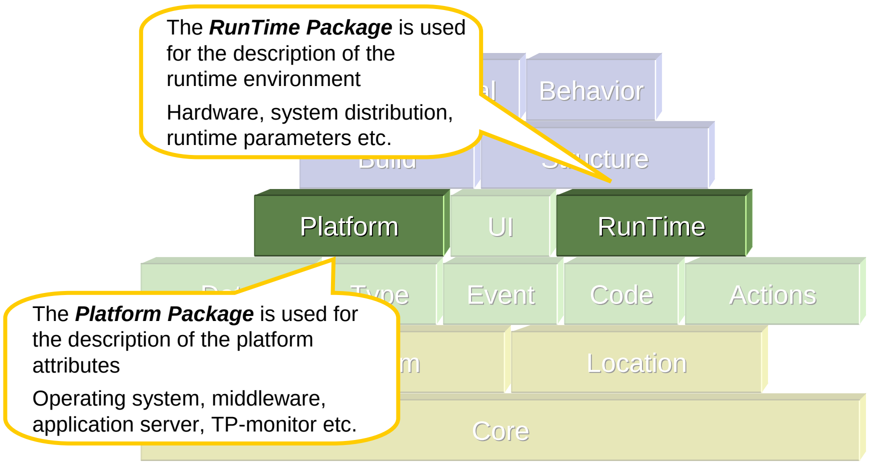
The ***UI Package*** is used for the description of all aspects of the user interfaces

Configuration, structure, content, sequence of the operations, relation to the infrastructure

The ***Event Package*** is used for the description of the triggering and processing of „events"

UI-events, messages, trigger

Platform & RunTime Package



The ***RunTime Package*** is used for the description of the runtime environment

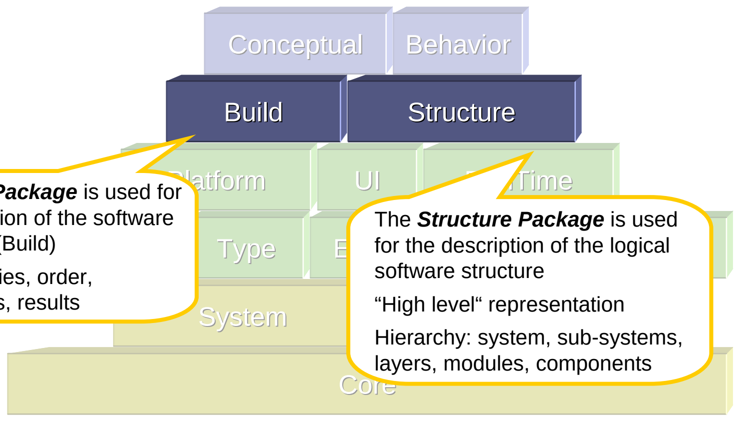
Hardware, system distribution, runtime parameters etc.

The ***Platform Package*** is used for the description of the platform attributes

Operating system, middleware, application server, TP-monitor etc.

DELTA software technology
The Generator Company

Build & Structure Package



The ***Build Package*** is used for the description of the software production (Build)

Dependencies, order, components, results

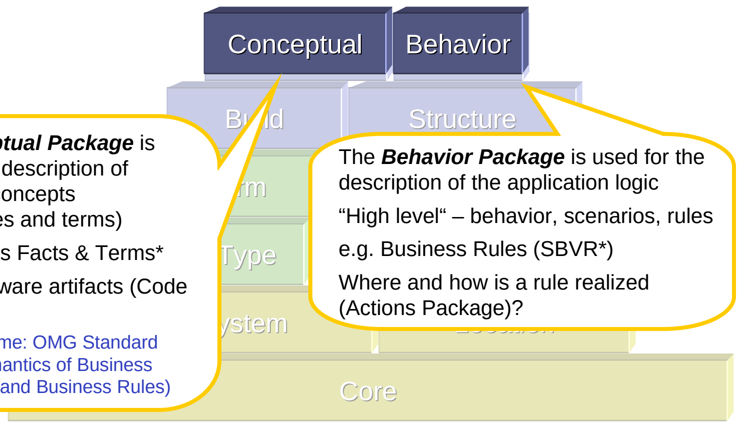
The ***Structure Package*** is used for the description of the logical software structure

“High level“ representation

Hierarchy: system, sub-systems, layers, modules, components

DELTA software technology
The Generator Company

Conceptual & Behavior Package



The ***Conceptual Package*** is used for the description of application concepts (i.e. Attributes and terms)

e.g. Business Facts & Terms*

Links to software artifacts (Code Package)

*soon to come: OMG Standard SBVR (Semantics of Business Vocabulary and Business Rules)

The ***Behavior Package*** is used for the description of the application logic

“High level“ – behavior, scenarios, rules

e.g. Business Rules (SBVR*)

Where and how is a rule realized (Actions Package)?

DELTA software technology
The Generator Company

Abstract Syntax Tree Meta-Model (ASTM)

- The following questions arise:
  - Where does the information about programs (and similar artefacts) in the KDM come from?
  - How are the details of the program code represented?
  - How can programs containing different syntaxes be analyzed together?
  - How can programs of one syntax be converted into another one?
- The answers to these and other questions require an appropriate and abstract representation
  - Usually, these are *Abstract Syntax Trees*
  - Many different kinds of ASTs already exist, e.g. within compilers
  - Anyway, the answer to the above questions requires a common meta-model:
    - *Abstract Syntax Tree Meta-Model*

DELTA software technology
The Generator Company

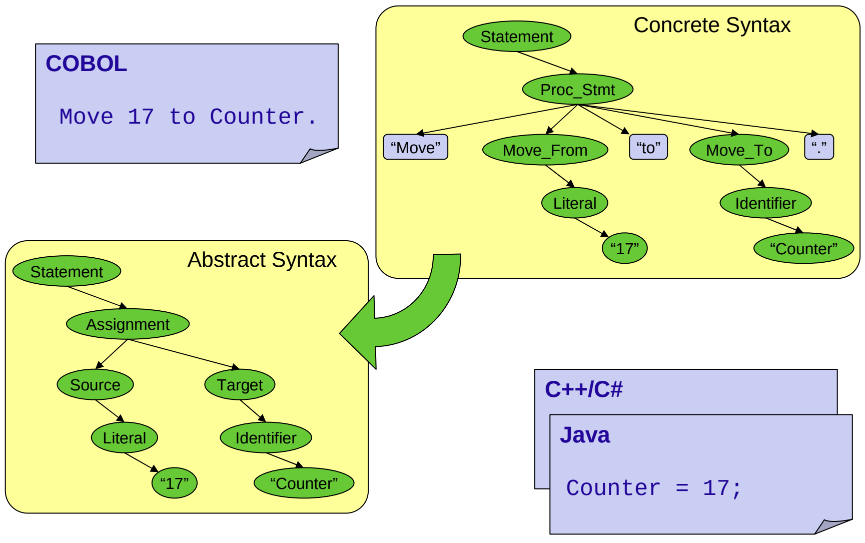
What Is An Abstract Syntax Tree?

- An AST is ...
- ... A formal representation of a software‘s syntax structure
- ... More suitable for formal analysis than the concrete syntax (*Surface Syntax*)
- ... A more precise formalism for the detailed acquisition of information than less formal techniques (scanner, tokenizer, visual examination)

Abstract Syntax Tree Abilities

- An AST …
- ... can allow the generation (or reproduction) of concrete syntax of (legacy) systems from the abstract syntax
- ... can be enriched by analysis results (interrelation, data flow, control flow etc.)
- ... can be used for deriving software metrics and documentation
- ... can be transformed into other AST models by using suitable transformation rules
- ... can be interpreted and manipulated by using model query and manipulation languages (e.g. OMG’s QVT™, TSRI’s JTGEN™/JRGen™ and ANGIE™ - a Delta tool)

Source Code And AST



COBOL

```
Move 17 to Counter.
```

C++/C#

Java

```
Counter = 17;
```

DELTA software technology
The Generator Company

GASTM, SASTM

- SASTM ➔ Specific AST Meta-Model ➔ PSM (MDA)
  - E.g. COBOL, C#, Java
- GASTM ➔ Generic AST Meta-Model ➔ PIM (MDA)
  - neutral, language-independent AST
- Transformation (e.g. C++ ⇔ Java)
  - C++-AST ⇔ Java-AST ☹
  - C++-AST ⇔ GAST ⇔ Java-AST ☺
  - MDA: PSM ⇔ PIM ⇔ PSM

DELTA software technology
The Generator Company

ASTM – The Standard

- Sorry to say that the OMG Task Force (ADMTF) is a little behind schedule at the moment
- There is currently no standard available or final common proposal
- What we already have are three different submissions which are most likely to be combined:
  - Discrete Model – The Software Revolution (TSRI)
  - Continuous Model – TCS Consulting
  - Interface Module Model – Klocwork
- A joint revised submission is scheduled for January 24th 2006

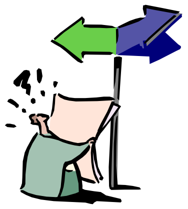
Discrete And Continuous ASTM

Discrete ASTM (TSRI)	Continuous ASTM (TCS)
GASTM is a comprehensive subset of all common language elements (of many languages)	
Each SASTM is based on an individual set of language elements for a specific language syntax	Each SASTM is an enhancement or specialization of the GASTM
SASTM/GASTM mapping rules provide the functional equivalence	SASTM combined with GASTM round off the AST model for each language
The analyses are completely executed on the GASTM basis	All analysis are performed on the base of the sum of GASTM and SASTM
Multi-language and language-neutral representations, analysis and transformations by GASTM and SASTM/GASTM mapping	Multi-language and language-neutral representations, analysis and transformations by means of the association model (GASTM + associated SASTMs)

ASTM Interface Model – Klocwork

- Interface definition instead of a specific ASTM
  - It is assumed that several ASTs already exist (e.g. within compilers)
  - The interfaces provide a common view
  - GASTM and SASTM are thus virtual
- Architecture similar to KDM
  - Core package (basic constructs)
  - Limited subset for GASTM
    - Scope, statement, expression, identifier package (explicit)
    - Define use, control, data flow (implicit)
  - “Light-Weight“ enhancement mechanisms

Architecture-Driven Modernization



- ☑ Modernization
- ☑ Scenarios
- ☑ ADM Task Force
- ☑ Standards
- ➢ ADM Now?

ADM Principles

- As the standards are not yet worked out in detail, it is not possible to have any hands-on experience
  - However, experience is available with the essential principles of ADM

1. Model-based
   - All information on the artefacts to be processed are stored in standardized models. This also applies to derived knowledge
   - Using MOF and XML (XMI) as the conceptual and technical base reduces the tool-dependency and allows third-party products to be used
   - It is only by defining meta-models that the use of automated discovery and analysis tools really becomes reasonable

ADM Principles

2. Abstraction and platform-independency
   - Primary sources are just the starting point for the population of the models that are to serve as the basis for all subsequent operations
   - Wherever possible, the derived knowledge is abstracted and represented in a platform-independent way
3. PSM-PIM-PSM transformation
   - This method has proved its value for all cases in which modernization deals with the creation or modification of software (migration, transformation, integration)
   - The significantly reduced complexity associated with this method increases transparency, reduces the error rate and therefore leads to higher efficiency
   - By the way, using this method makes platform decisions less crucial because changing the definition of the target platform is rather easy

ADM Principles

4. Target-oriented and selective
   - “The more the better” surely does not apply to the effective support of modernization measures
   - Meta-data are to be collected and analyzed as target-oriented as possible
5. Adaptive “Factory”
   - Modernization tasks are so diverse that there is no out-of-the-box “all singing and dancing” solution
   - Likewise, you cannot expect a specific tool for each modernization scenario
   - What is really needed are tool components that are based on one common model and that can be assembled into a specific modernization factory

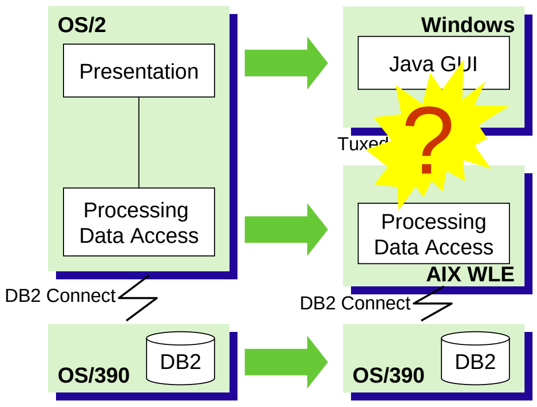
Modernization in Practice I

- The first example was finished about three years ago but it already contained fundamental elements of the ADM principles
  - It is already **model-based** though not KDM or ASTM (they did not exist at that time)
  - The deployed model is based on **MOF** and **XML**
  - The principle of **platform-independency** was already established in this model
  - As well as the **PSM-PIM-PSM** principle
  - The **selectivity** principle particularly applies in this case because almost everything focuses on interfaces and communication

Modernization in Practice I

- This task is similar to the scenarios IV (Platform Migration) und V (Non-Invasive Application Integration)
- Starting point:
  - COBOL application with fat clients (MF Dialog System) running on the obsolete platform OS/2
  - 3,600 program modules
  - 410 Screens
  - 5,600 copybooks (→ interface definition)
  - > 2,000 terminals
- Objective:
  - A real 3-tier-architecture with thin clients (Java)
  - Application server on Unix (BEA WebLogic Enterprise)
  - Communication via Tuxedo

From Architecture Mismatch ...



➔ Data formats

➔ Interfaces

⇨ Classes/methods vs. Call interface

➔ Granularity

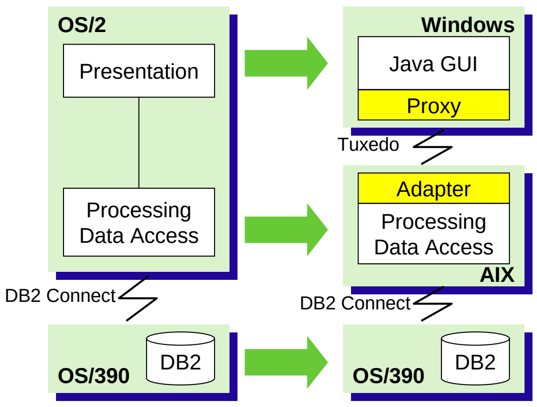
⇨ COBOL structures > 120KB

➔ Transaction logic

➔ Client framework

➔ . . .

... To An Adaptive Architecture



➔ Proxy

⇨ Real Java classes and formats

⇨ Framework compatible

➔ Adapter

⇨ Mapping call interface ⇔ operations/parameters

⇨ State management

⇨ Transaction adaptation

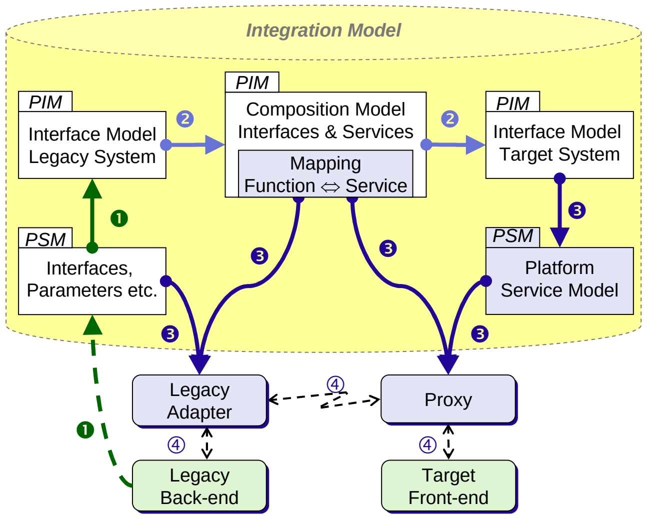
➔ Test driver, log files

➔ . . .

Integration With Triple Jump

- Discovery
  - Import of program interfaces whose parameter structures in this case were provided as COBOL copybooks
  - Automatic transformation into a platform-independent model representation – MOF- and XML-bases – **similar to KDM**
  - This scenario did not require any further information (**selectivity**)
- Composition
  - Definition of the target model: interfaces, operations, parameters
  - Mapping definition: old model ⇔ new model
  - ➔ *partially processed automatically, software support for the remaining tasks, declarative and platform-neutral*
- Production
  - Complete generation of adapters and proxies (COBOL and Java)
  - Complete generation of the mapping routines (COBOL)
  - Complete generation of the communication layer (Tuxedo)
  - Just a few manual interventions within the old modules

PSM→PIM→PSM For The Integration


Integration Model
PIM
Interface Model Legacy System
PIM
Composition Model Interfaces & Services
Mapping Function ⇔ Service
PIM
Interface Model Target System
PSM
Interfaces, Parameters etc.
PSM
Platform Service Model
Legacy Adapter
Proxy
Legacy Back-end
Target Front-end
❶ ❷ ❸ ④

Integration of a legacy system with a new target platform

❶ *Discovery*
i.e. Extract from source

❷ *Composition*

❸ *Production*
i.e. transformation with generators

④ *Runtime*

The Result

- The application was put into production long ago and fulfilled all expectations
- The cost projection and the deadline for the overall project were achieved ...
- ... even though the manual conversion of the user interface (Java) turned out to require much more effort than estimated
- This was compensated for by the automated transformation and integration of the COBOL modules using the SCORE® tools
- ***According to the customer – Five times faster than a manual solution***

Modernization in Practice II

- The second example is up-to-date and was prepared in parallel to the development of the first ADM standards
- It thus not only comprises ADM principles but also preliminary implementations of the KDM and ASTM standards
- High demands on quality and security

  together with some technical particularities

  require a very individual modernization factory

Modernization in Practice II

- Involves tasks from scenarios II (Application Improvement), III (Language-to-Language Conversion) and IV (Platform Migration)
  - This is the most frequent combination of tasks where large platform changes are involved
- Starting point:
  - COBOL applications with online and batch programs on Unisys OS2200 with RDMS (call interface not embedded SQL)
  - Delta ADS generator (Batch), but without Delta online and database support which would have made a migration easier
  - 1,500 online programs
  - 3,500 batch programs
  - 1,500 screens
  - 800 copybooks and macros

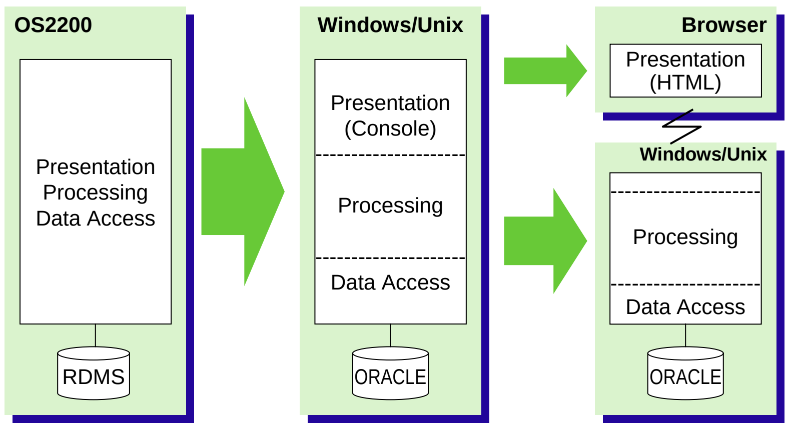
Modernization in Practice II

- Strategic objectives:
  - Cost reduction of the production environment
  - Break out of existing (and future) proprietary constraints through platform-independency!
- Technical objectives:
  - Complete replacement of the Unisys 2200 platform
  - Target platform Windows or Unix with ORACLE etc.
    - At the beginning of the project the final platform has not yet been chosen!
  - Replacement of all OS2200 specific constructs (COBOL dialect, DBMS, Screen)
  - Presentation of a Web perspective

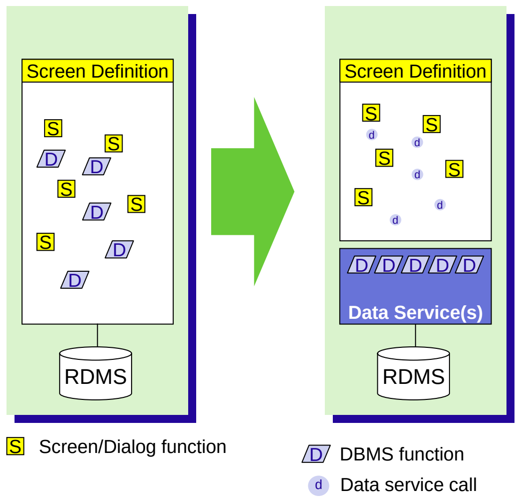
Modernization in Practice II

- Project terms of reference:
  - Risk control and risk minimization – any production interruptions or malfunctions will not be accepted
  - Non-blocking – development and maintenance can only be interrupted and frozen on a short-term basis, and only ever for some components in parallel and never for the complete application
  - Automation, reproducibility and traceability
  - The efficiency of software development which is currently supported by perfected software development processes must not be impacted at all

The Objective



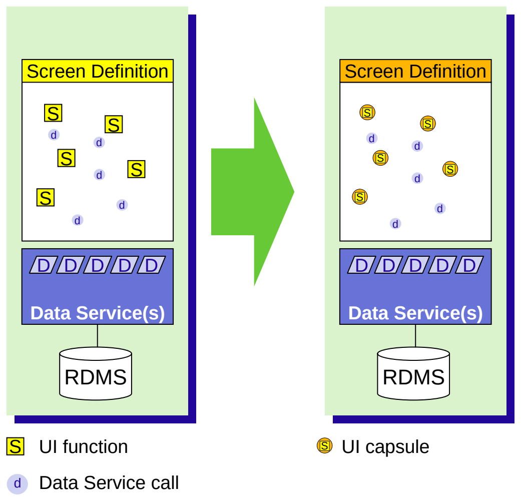
Outsourcing Data Access



- Data Service components
  - Collecting and combining DBMS functions
  - Outsourcing them into separate *Data Service* component(s)
  - Replacing DBMS function by a service call
- Objective
  - Isolation of data functions
  - Simplifying migration, test and verification
  - Temporary solutions needed for the migration will not be part of the application "for ever and a day"

➔ Prerequisite for the *In-Place Migration*

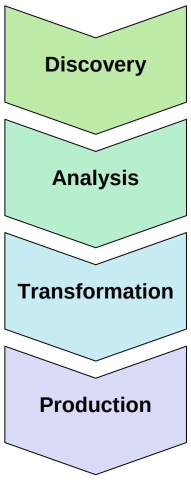
Encapsulation Of The User Interface



- UI capsules
  - UI functions cannot be outsourced without changing the application functions
  - They are encapsulated within (platform-independent) macro calls
- Objective
  - Isolation of the UI functions
  - Similar to DBMS functions

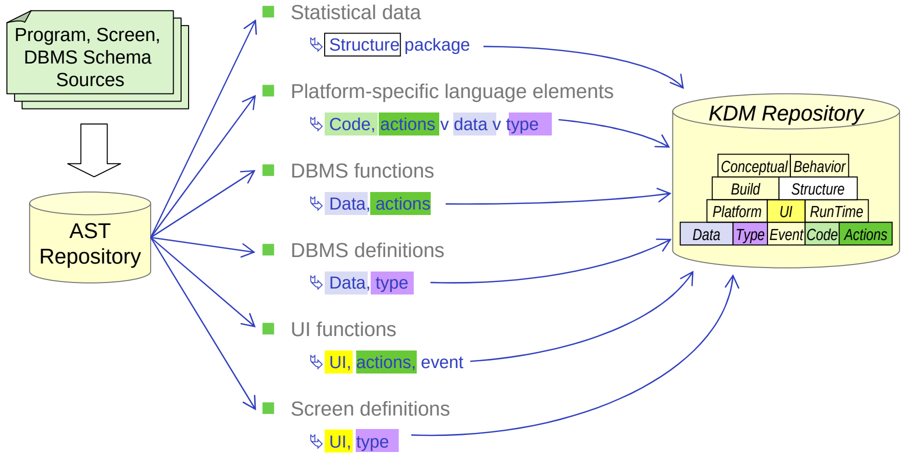
➔ Prerequisite for *In-Place Migration*

Assembly Line Of The Modernization Factory



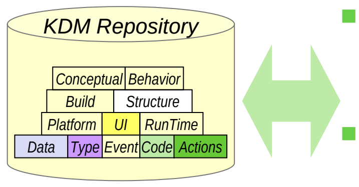
- Discovery
  - Import the sources into an AST repository
  - Integrate the relevant data into the KDM repository
- Analysis
  - Statistical evaluation and metrics
  - Analysis of the transformation candidates
- Transformation
  - Converting platform-specific constructs into platform-independent ones
- Production
  - Generation of the target artefacts (sources etc.)

Populating The KDM Repository



- Statistical data
  - ⇘ Structure package
- Platform-specific language elements
  - ⇘ Code, actions v data v type
- DBMS functions
  - ⇘ Data, actions
- DBMS definitions
  - ⇘ Data, type
- UI functions
  - ⇘ UI, actions, event
- Screen definitions
  - ⇘ UI, type

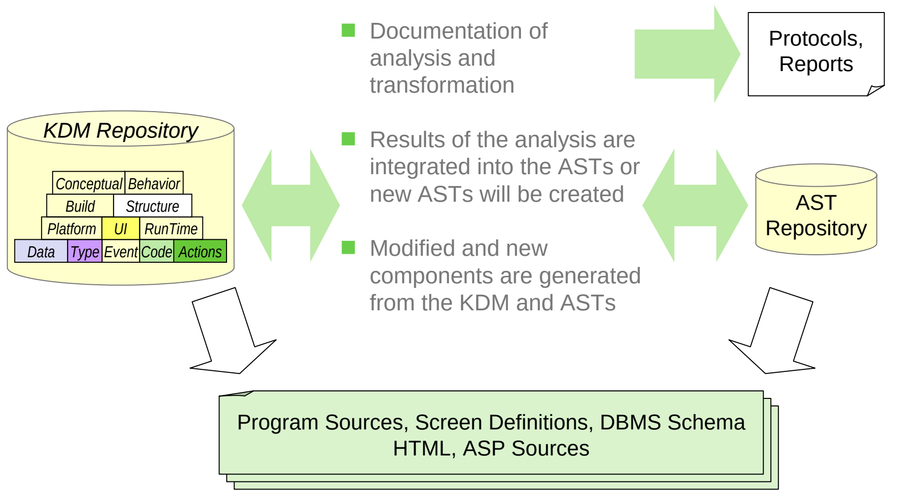
KDM Analysis And Transformation



- Analyzing platform-specific language elements
  - In: Code, In/Out actions v data v type
- Analysis, grouping and transformation* of DBMS definitions and functions
  - In: data, type, actions, Out: data, actions
- Analysis, grouping and transformation * of screen definitions and UI functions
  - In: type, In/Out: UI, actions, event
- Analysis- and transformation rules will partially be defined (or further developed) during the course of the project

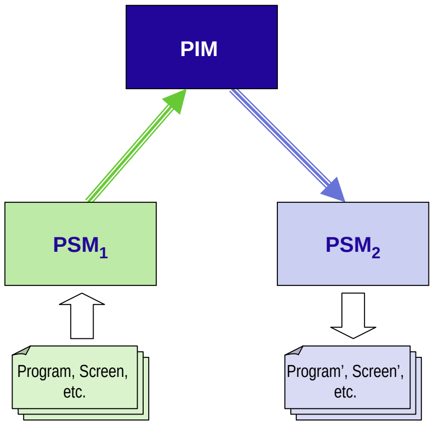
•Transformation: Translation of platform-specific constructs into platform-independent ones

Production



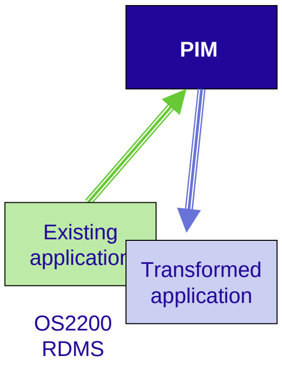
- Documentation of analysis and transformation
- Results of the analysis are integrated into the ASTs or new ASTs will be created
- Modified and new components are generated from the KDM and ASTs

PSM→PIM→PSM Transformation



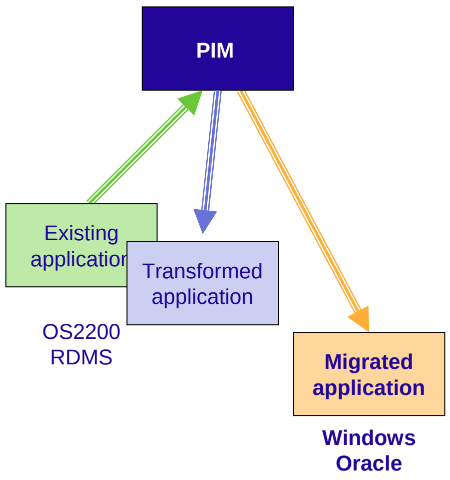
- PSM$_1$
  - Source artefacts (program sources etc.), are automatically imported
  - ASTM and KDM
- PIM
  - Results from analysis and transformation
  - Platform-neutral model of the transformation-relevant elements
  - KDM
- PSM$_2$
  - Target artefacts
  - Platform-specific conversion
  - ASTM and KDM

First Step: “In-Place” Migration



- After the analysis and transformation the application will be generated “back” to the original platform
  - With unchanged interfaces and functions
  - Built on a new architecture – Data Services and UI capsules,
  - Instrumented by test aids
  - Clusters of any size
- Test within the usual maintenance environment
  - Utmost security due to the established test environment
  - Large or small increments at will
  - Reversible
- No “Big-Bang”, no “No-Return”

Second Step: Final Migration



- Final platform – transformed and tested programs are once again generated, compiled etc.
- Automated regression tests derived from the first step
- Short cycle time

  ➔ production interruption reduced to the minimum
- Utmost security

  ➔ no risks, no (nasty) surprises

Succeeded?

	Objective	Solution
✓	Risk control and risk minimization – any production interruptions or malfunctions will not be accepted	Model-based approach offers continuous control of the actions, impact analysis etc. 2-step-migration and automated test reduce the risk to a minimum
✓	Non-blocking – development and maintenance can only be interrupted and frozen on a short-term basis, and only ever for some components in parallel and never for the complete application	*In-Place* migration and automation guarantee short idle periods for a manageable number of application components during each migration step
✓	Automation, reproducibility and traceability	All process-relevant information is stored within the models and can thus be traced back to the source – KDM principle Automation is the basis for all tools
✓	The efficiency of software development which is currently supported by perfected software development processes must not be impacted at all	The extendable layer architecture even increases the efficiency

DELTA software technology
The Generator Company

Succeeded?

- Objective – cost reduction of the production ✓
  - Succeeded! – by using more cost-efficient platforms
  - The migration does not affect the options
- Objective – to break existing (and future) proprietary constraints through platform-independency ✓
  - Concerning the language (COBOL) all OS2200 elements are replaced by standard elements.
  - Data and user interfaces are encapsulated and platform-independently stored within models. According to the MDA paradigm, the platform-specific components are then generated from the model.
  - If this is no longer required, the generated artefacts can be maintained without any tools.

DELTA software technology
The Generator Company

Architecture-Driven Modernization

☑ Modernization?

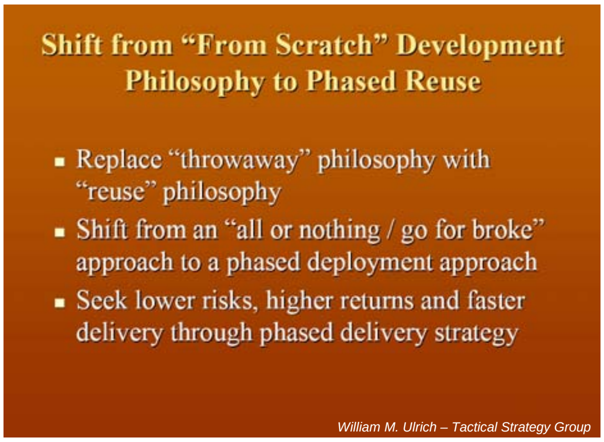
☑ Scenarios

☑ ADM Task Force

☑ Standards

☑ ADM Now~~?~~!

ADM Now!



Shift from "From Scratch" Development Philosophy to Phased Reuse

- Replace "throwaway" philosophy with "reuse" philosophy
- Shift from an "all or nothing / go for broke" approach to a phased deployment approach
- Seek lower risks, higher returns and faster delivery through phased delivery strategy

William M. Ulrich – Tactical Strategy Group

- **Modernization** is not the only but an **indispensable alternative** when increasingly demanding **business requirements** are to be realized
- **Standards prevent** cost-intensive **uncontrolled growth**, **provide paradigms** and **interoperability** between the tools of different vendors

ADM Now!

- ADM provides perfect strategies for modernization
- The instruments for successful modernization projects are already raring to go – event though there is still some work on the standards to be done:
  - Models – already defined by the standard
  - Methods – the basis is already given by the ADM principles, additional ones will be defined successively in the course of time
  - Tools – up-to-date tools already implement the ADM concepts
- Let's get started ...
  - ... On-going development for these and other OMG standards will provide an even better support for your projects on a step-by-step basis, as well as leading to an increasing variety of tools and vendors

Delta's Modernization Tools

- SCORE® Transformation Factory
  - Model-based: MOF, XML, following the ADM standards
  - Components for discovery, analysis, transformation and generation
  - Configurable and extendable to individual demands
- SCORE® Adaptive Bridges
  - Generative Service Enablement
  - Model-based: MOF, XML, MDA, EDOC
  - Non-invasive
  - Platform-independent, bridges technologies

Architecture-Driven Modernization

Thank you for your attention

- Additional information

  www.D-S-T-G.com

  adm.omg.org
  www.omg.org

  www.systemtransformation.com
  www.klocwork.com
  www.softwarerevolution.com

Wiliam M. Ulrich: Legacy Systems: Transformation Strategies, Prentice Hall, 2002