DELTA software technology

# Software-Modernisierung und -Migration automatisieren mit

# AMELIO

## Modernization Platform

## Fragen und Antworten

# ÜBERSICHT

# WELCHE MIGRATIONSMETHODIK LIEGT AMELIO ZUGRUNDE?

***Die Methodik der Anwendungsmigration mit AMELIO ist zu 100% werkzeugbasiert, nicht nur ein Leitfaden und nicht durch manuelle Abläufe gesteuert.***

Die AMELIO-Methode für die Migrationen und Modernisierungen bietet die vollständig automatisierte Durchführung aller Änderungen. Sie reduziert sich damit nicht auf eine Verfahrens- und Vorgehensbeschreibung, ist also mehr als ein Benutzer- oder Referenzhandbuch.

Das Prozess-Modell ist zu 100% werkzeugbasiert.

- Eine konsequente Methode,
- werkzeugbasiert bis runter zu den kleinsten Details,
- ohne dass irgendein Bypass notwendig, möglich oder erlaubt ist.

Die Methode (das Clean Room-Konzept) ist vollständig in die Tools implementiert.

# EMPFOHLENE STRATEGIEN FÜR MODERNISIERUNGEN UND MIGRATIONEN

***Aufgrund der verfolgten modellgetriebenen Ansätze und der zur AMELIO Modernization Platform gehörenden Komponenten sind für Modernisierungen unterschiedliche Strategien anwendbar. Sie reichen vom klassischen Big Bang bis zu Mehr-Schrittansätzen, die auch In-place-Migrationen einbeziehen können.***

AMELIO Modernization Platform setzt die modellgetriebenen Konzepte der Architecture Driven Modernization (ADM) der OMG um.

Die zu ändernden Komponenten werden in Knowledge Models überführt (Discovery), auf denen anschließend die Analysen und Transformationen durchgeführt werden.

Die Sauberkeit dieses konzeptionellen Ansatzes erlaubt verschiedenste Migrationsstrategien. AMELIO ist nicht auf einen einzigen Ansatz oder auf eine einzige Vorgehensweise fixiert.

Neben dem am häufigsten anzutreffenden „1-Schritt“-Vorgehen („Big Bang“ oder „One-off Migration“) sehen wir im „Mehr-Schritt“-Verfahren mit seinem „sanften“ Vorgehen („smooth transition“) eine bedenkenswerte Strategie.

Mehr-Schritt-Verfahren scheinen zunächst über Umwege zum Ziel zu führen. Sie erleichtern jedoch die Umsetzung komplexer Modernisierungsaufgaben, wie z.B. Architekturtransformation oder Plattformmigrationen, liefern frühzeitig Ergebnisse und reduzieren die Risiken.

Detailliertere Beschreibungen der 1-Schritt- und Mehr-Schritt-Verfahren finden sich in eigenen Beiträgen.

# WIE SEHEN MODERNISIERUNGS- UND MIGRATIONSKONZEPTE BEIM EINSATZ VON AMELIO AUS?

***AMELIOs Modernisierungskonzepte unterscheiden sich grundsätzlich von allen manuellen oder halbautomatischen Ansätzen.***

***Mit den Bausteinen der AMELIO-Plattform wird eine Fertigungsstraße für Modernisierungsaufgaben konfiguriert, die präzise die Aufgaben des jeweiligen Projekts erfüllt.***

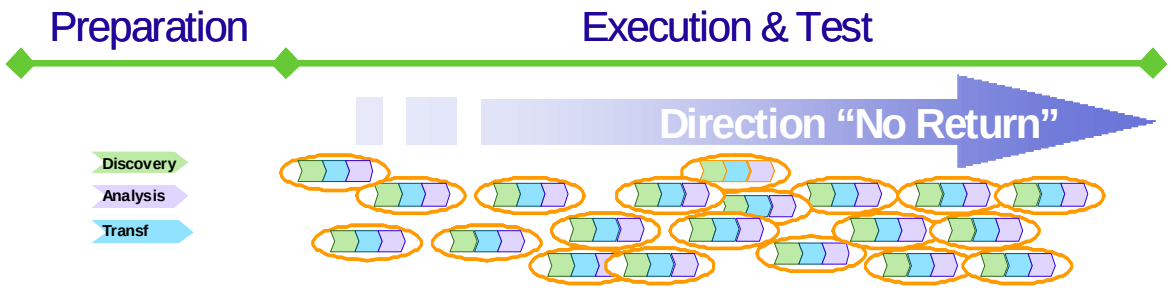
***Durch diese Spezialisierung, den modellgetriebenen Ansatz und das Prinzip des Meta-Level-Testens ist es möglich, eine 100%-ige Automation mit einer Fehlerquote nahe 0% zu erzielen.***

Der automatisierte Lösungsansatz mit AMELIO Modernization Platform unterscheidet sich grundsätzlich von manuellen oder halbautomatischen Vorgehensweisen.

Jede Form von Modernisierung und Migration (manuell, halbautomatisch, automatisch) erfolgt in 3 wesentlichen Schritten:

- Discovery
    - Informationen aus allen Sourcen und anderen Quellen sammeln oder/und importieren
    - Identifikation der Änderungskandidaten
- Analyse
    - Überprüfung der Änderungskandidaten
      Ermittlung der notwendigen Änderungen und deren Auswirkungen
    - Aufspüren von Widersprüchen
- Transformation
    - Ausführung der Änderungen, Bereitstellung der modifizierten Komponenten

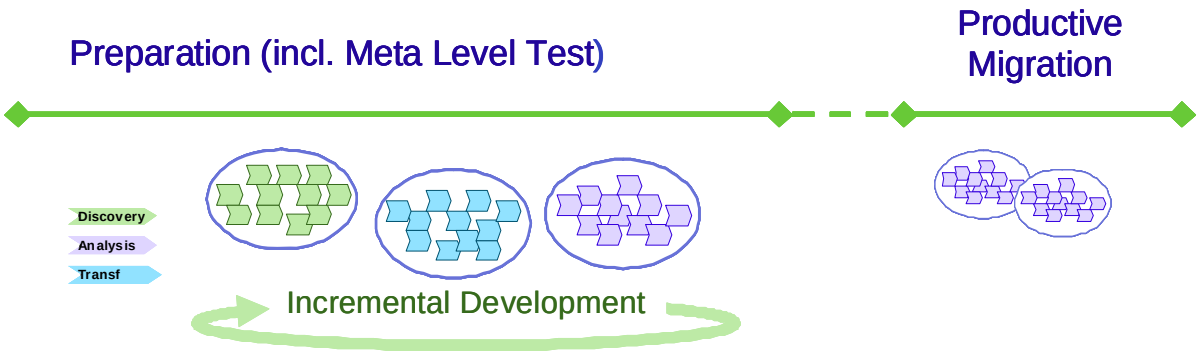
## Manuelles Vorgehen




Eigenschaften manueller Modernisierungen:

- Nach einer mehr oder weniger kurzen Vorbereitungszeit …
- … wird eine Komponente nach der anderen untersucht (Discovery), analysiert und transformiert
- … mit Stunden oder Tagen Aufwand.
- Nur eine kleine Anzahl Module kann von einer Person parallel bearbeitet werden.
- Andauernde Tests während der gesamten Modernisierung
- Am Schluss wird die transformierte Anwendung mit einem „Big Bang" in Produktion genommen.
  Im Fall eines Plattformwechsels wird produktiv auf die neue Plattform gewechselt, nachdem auch die Daten auf die neue Plattform migriert wurden.
- Lange Blockadezeit („freezing time")
- Skalierung des Projekts über Anzahl Mitarbeiter

## Der AMELIO-Ansatz




Eigenschaften der automatisierten Modernisierung mit AMELIO Modernization Platform:

- Vorbereitung
  - Das Anwendungssystem wird komplett als Ganzes untersucht (Discovery), im Zusammenhang mit dem gesamten System analysiert und anschließend automatisch transformiert

    - Meta-Level-Tests werden für präzise ermittelte Test-Sets durchgeführt.
    - Neue Erkenntnisse aus Analysen und Tests werden durch Anpassungen der Fertigungsstraße umgesetzt
    - Der Kreislauf von
      -- Discovery
      -- Analyse
      -- und Transformation
      kann leicht mit nur geringem Aufwand wiederholt werden.
- Produktive Transformation
    - Cluster beliebiger Größe werden auf Anforderung transformiert
    - … mit einem Aufwand von nur wenigen Minuten, allenfalls Stunden (pro Cluster!)
      Dadurch sehr kurze Blockadezeit.
    - Hierfür werden nur wenige Mitarbeiter benötigt.
    - Skalierung des Projekts über Hardware

Die Hauptphasen Discovery, Analyse und Transformation sind sauber voneinander getrennt und werden entsprechend den Projektanforderungen terminiert.

Discovery und Analyse werden getrennt von der Transformation vorneweg durchgeführt. Sie blockieren keine parallel durchgeführten Entwicklungs- oder Wartungsprojekte.

Die produktive Transformation großer Modulstückzahlen erfolgt zum Abschluss und benötigt nur ein kurzes Zeitfenster, verursacht daher auch nur eine kurze Blockadezeit („freezing time“).

In dieser Prozessabfolge gibt es keinen „point of no return“. Wenn notwendig, können Anpassungen an der Fertigungsstraße vorgenommen und die Prozesse jederzeit mühelos wiederholt werden.

# MIGRATIONSSTRATEGIEN: BIG BANG ODER EIN-SCHRITT-VERFAHREN

***Die Ein-Schritt-Migration ist ein Big Bang-Ansatz.***

***Sie ist auf den ersten Blick mit weniger Aufwand verbunden, birgt wegen der üblicherweise großen Anzahl gleichzeitig zu lösender Aufgaben aber ein größeres Risiko in sich.***

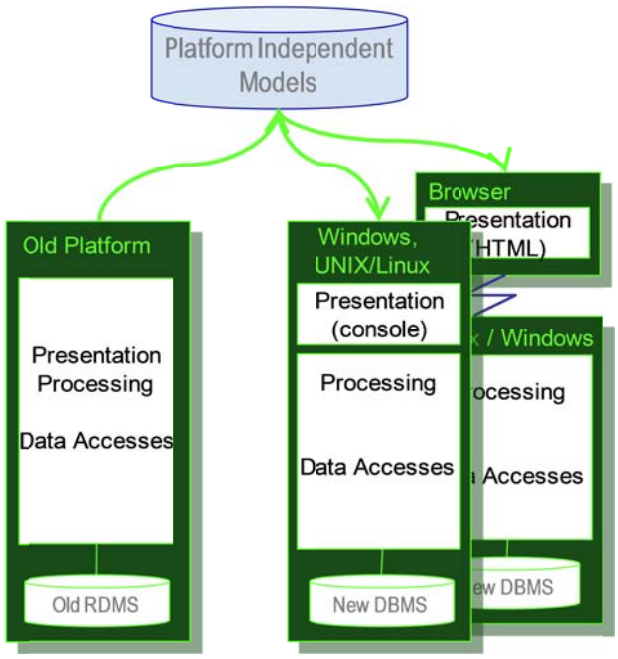
***Um diesen Ansatz erfolgreich durchführen zu können, müssen die für die Modernisierung vorgenommenen Änderungen allerhöchste Qualitätsansprüche erfüllen.***

***Zusätzlich muss eine kürzest mögliche Blockadezeit erreicht werden.***

Im Falle eines Plattformwechsels wird ein Ein-Schritt-Verfahren für die bestehenden Komponenten

- die Abhängigkeiten von der alten Plattform,
- die Datenzugriffe auf das alte DBMS,
- und die alten Benutzer-Frontends

in einem einzigen Transformationsschritt ersetzen.




Die in plattformunabhängigen Modellen gespeicherten Informationen werden genutzt, um Code-Module für die neue Plattform (z.B. Microsoft Windows, UNIX/Linux) zu erzeugen.

- Alle in den Komponenten enthaltenen Spezifika der alten Plattform werden ersetzt durch Gleichwertiges der neuen Plattform.

- Datenzugriffe auf das alte DBMS werden ersetzt durch Zugriffe auf das neue DBMS. Im Rahmen einer Architekturtransformation werden sie sogar aus den Programmen in separate Daten-Services ausgelagert.
- Für die Ablösung klassischer „Green Screens“ und des Masken-Handlings sind zwei Ansätze vorstellbar:
    - Einführung eines „Green Screen“-Äquivalents
    - Wechsel zu Browser-basierten Frontends mit Java/C#-Implementierung

Zusätzlich zu den Code-Änderungen für den Plattformwechsel können Verbesserungen an den Anwendungen vorgenommen werden.

Solch eine Modernisierungslösung mit AMELIO ist eine No-footprint-Lösung, d.h. keine zusätzlichen Produkte werden durch die Transformationslösung hinzugefügt.

# MIGRATIONSSTRATEGIEN: MEHR-SCHRITT-VERFAHREN

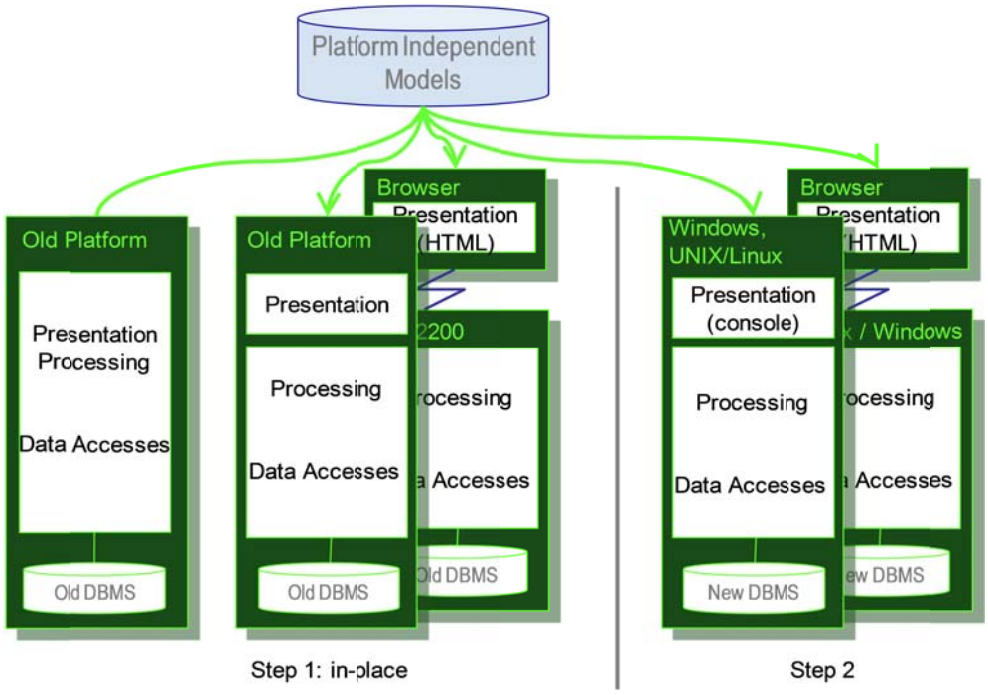
***Mehr-Schritt-Verfahren ermöglichen sanfte Änderungs- und Modernisierungsprozesse. Die Gesamtaufgabe wird dazu in voneinander unabhängige Teilaufgaben zerlegt, die dann eine nach der anderen gelöst werden.***

***Jeder Teilschritt liefert ein in sich geschlossenes Ergebnis, das für sich getestet und sogar produktiv genutzt werden kann.***

***Der erforderliche Aufwand erscheint auf den ersten Blick größer als bei einem Big Bang-Ansatz, dies kann aber leicht durch Einsatz effizienter Automatisierungs-techniken ausgeglichen werden.***

***Das entscheidende „Pro-Argument" ist die starke Reduktion der mit der Komplexität der Gesamtaufgabe verbundenen Risiken.***

Ein Plattformwechsel von einer Legacy-Plattform zu einer Microsoft Windows- oder UNIX/Linux-Plattform kann z.B. in zwei kleineren, leichter zu beherrschenden Schritten durchgeführt werden.




Im ersten Schritt – der „In-place"-Migration – ist das Migrationsziel die bestehende Legacy-Plattform: Die modifizierten Anwendungen werden nach wie vor auf der alten Plattform ausgeführt. Dabei können dann sogar transformierte Programme zusammen mit nicht transformierten ausgeführt werden.

Im zweiten Schritt werden die geänderten Anwendungen dann für die neue Plattform bereitgestellt.

Inhalte und Ergebnisse des In-place-Migrationsschritts:

- Die Spezifika der alten Plattform werden ersetzt durch plattformneutralen Code oder in einem separaten Layer isoliert.
- Die bestehenden Datenzugriffe auf das alte DBMS werden neutralisiert oder für das alte DBMS werden Daten-Services bereitgestellt.
- Auch für die Ersetzung alter Benutzer-Frontends und des zugehörigen Codings gibt es zwei Alternativen:
    - Einführung eines „Green Screen"-Äquivalents für die neue Plattform
    - Browser-basiertes Frontend mit Java-/C#-Implementation

Diese 2-Schritt-Migration mit AMELIO ist eine No-footprint-Lösung – genau wie die an anderer Stelle beschriebene Ein-Schritt-Migration. Das bedeutet: es werden durch die Plattform-Migration keine zusätzlichen Produkte eingeführt.

Der skizzierte 2-Schritt-Ansatz erlaubt Tests der neutralisierten Anwendungen, die schon für die neue Plattform vorbereitet sind, in der alten gut bekannten und beherrschten Umgebung. Testvergleiche der originalen und neuen Programme sind in dieser Umgebung leicht möglich.

Sollen Modernisierungen und / oder funktionale Verbesserungen zusätzlich zusammen mit den Änderungen für den Plattformwechsel durchgeführt werden, so stehen diese schon auf der alten Plattform zur Verfügung.

# WELCHE BEDEUTUNG HAT DIE IN-PLACE-MIGRATION IN MIGRATIONSPROJEKTEN?

***Bei der In-place-Migration werden wesentliche Transformationen durchgeführt, die transformierten Anwendungen werden aber zunächst auf der Ursprungsplattform belassen.***
***Entweder ist der Verbleib auf der Ursprungsplattform Teil der Migrationsaufgabe oder aber ein Zwischenschritt eines sanften Übergangs.***

Bei einer In-place-Migration werden umfangreiche Migrationsänderungen durchgeführt, die geänderten Anwendungen werden aber weiter auf der ursprünglichen Plattform eingesetzt.

Hierbei können wir zwei verschiedene Migrationsaufgaben unterscheiden, die beide auf den In-place-Ansatz zurückgreifen:

- Die Ergebnisse einer Architekturtransformation können aufgrund der implementierten Lösung auf der Ursprungsplattform eingesetzt werden, genauso auf einer neuen Plattform.
- Ein Plattformwechsel soll über einen Zwischenschritt realisiert werden, der zunächst Plattformunabhängigkeit herstellt.

Gemeinsam ist beiden Aufgabenstellungen, dass die Lösungen auf generative Ansätze zurückgreifen.

Im ersten Fall werden von der AMELIO-Fertigungsstraße Architekturänderungen durchgeführt, wie z.B. Auslagerung von Datenzugriffen in Data Services, weitergehende Einführung einer Schichtenarchitektur, allgemeines Service Enablement. Die Transformationsergebnisse können bereits auf der alten Plattform getestet und produktiv eingesetzt werden.

Die generativen Komponenten der AMELIO-Fertigungsstraße erzeugen die geänderten Code-Module. Wenn die transformierten und in-place bereits getesteten Anwendungen nach ihrem In-place-Einsatz auf eine neue Plattform portiert werden sollen, müssen die Generatoren nur für die Unterstützung der technischen Parameter der neuen Plattform sorgen.

Im zweiten, oben genannten Szenarium einer In-place-Migration ist für die Migration auf eine neue Plattform die Plattformunabhängigkeit als Zwischenziel definiert. Hierfür werden in den Source-Modulen alle Plattformabhängigkeiten durch plattformunabhängigen Code (z.B. nicht-prozedurale Deklarationen) ersetzt.

Dem Prinzip „single source – multiple targets“ folgend, werden die transformierten Source-Module zunächst für den In-place-Fall produziert, d.h. für die alte Plattform, und später dann für die neue Plattform. Auch in diesem Fall können in-place-transformierte und noch nicht transformierte Anwendungen kooperieren.

Falls die Migrations- und Modernisierungsaufgabe Änderungen an Schnittstellen erfordert, werden die Schnittstelleninkompatibilitäten zwischen migrierten und nicht-migrierten Anwendungen durch Fassaden überbrückt, die ebenfalls von AMELIO automatisch bereit gestellt werden.

# IN WELCHEN PHASEN LÄUFT EINE MIGRATION MIT AMELIO AB?

***Der AMELIO-Ansatz ist gekennzeichnet dadurch, dass das Projekt in kleine Abschnitte mit wohldefinierten Anfangs- und Endpunkten unterteilt wird, und dass die Prozesse vollständig automatisiert ohne manuelle Eingriffe ablaufen.***

Wir wollen uns hier auf die Betrachtung einer 1:1-Migration (also ohne weitergehende Modernisierungen) beschränken.

Unsere Erfahrungen haben uns gelehrt, Projekte in kleinere, leichter handhabbare Schritte zu unterteilen, von denen jeder einen wohldefinierten Anfangs- und Endpunkt mit eigenen Ergebnissen hat.

AMELIO-Fertigungsstraßen steuern folgende Prozesse:

i) Verstehen der heutigen Anwendung und der Zielvorgabe:
Identifizierung der Migrations- und Modernisierungsthemen und –probleme (d.h. der Dinge, die transformiert werden müssen) und erster Entwurf ihrer Transformationen

ii) Code-Import aller Source-Module in die AMELIO Knowledge Base

iii) Coverage-Analyse zur Ermittlung eines minimalen Test-Sets für das Meta-Level-Testing

iv) Design von Anwendungsarchitektur/Framework der neuen Plattform
Dies ist Voraussetzung für die endgültige Definition der Transformationsregeln

v) Konfiguration der Transformations- und Produktionsregeln

vi) Meta-Level-Testing der Factory

vii) Automatische Herleitung der Datenmigrationsprogramme (speziell für klassische Files)

viii) Cluster-weises Freezing und produktive Migration

ix) Eingeschränkte Roll-out-Tests, Übernahme der migrierten Anwendungen in die Produktion

Diese Prozesse werden von der konfigurierten AMELIO-Fertigungsstraße vollautomatisch ausgeführt, ohne dass manuelle Eingriffe möglich oder auch nur nötig wären.

# WAS WIRD MIT AMELIO AUTOMATISCH MIGRIERT UND WAS MUSS MANUELL MIGRIERT WERDEN?

***Wir sind der Ansicht, dass nur eine 100%-ige Automatisierung aller für eine Modernisierung notwendigen Änderungen eine akzeptable Qualität liefert. Als Konsequenz führt dies dann zu 0% manuellen Änderungen.***

Für Modernisierungs- und Migrationsaufgaben kommt nach unserer Überzeugung nur eine 100%-ige Automatisierung der notwendigen Massenänderungen und auch Architekturtransformationen in Frage.

In einem Massenänderungsprojekt kann es natürlich auch einzelne Programme oder extrem spezifische Modernisierungs-/Migrationsaufgaben mit nur einer sehr geringen Häufigkeit geben. Für diese Sonderfälle kann es kostengünstiger sein, sie von Hand zu transformieren, anstatt für diese Sonderfälle mit unverhältnismäßig großem Aufwand eine automatische Transformation innerhalb der Fertigungsstraße bereitzustellen. Dies ist dann ein Trade-off von Kosten und Nutzen.

# FÜGT AMELIO IRGENDWELCHE KOMPONENTEN HINZU, UM DAS VERHALTEN DES LEGACY-SYSTEMS ZU UNTERSTÜTZEN ODER ZU SIMULIEREN?

***AMELIO Modernization Platform ist eine „No vendor-lock-in“-/“No-footprint”-Lösung. Bei einer Migration oder Modernisierung werden keine neuen Laufzeit-komponenten eingeführt, die eine neue Abhängigkeit schaffen würden.***

AMELIO ist eine wirkliche „No-footprint“-Lösung. Sie produziert als Ergebnis Programme, die voll auf die neue Plattform ausgerichtet sind.

Um die migrierten Anwendung auf der neuen Plattform produktiv zu nutzen, werden keine herstellerspezifischen Produkte benötigt.

Für die Modernisierung oder Migration der Online-Umgebung bietet unsere Lösung weitreichende Möglichkeiten. Verschiedene Optionen stehen für die Ersetzung „klassischer“ Green Screens bereit:

- Browser-basierte Frontends mit native C#-/Java-Implementationen
- Ein „Green Screen“-Äquivalent mit ein paar als Source-Code bereitgestellten Lauf-zeit-Komponenten.

# KÖNNEN BEI EINER AUTOMATISIERTEN MIGRATION NEBEN MODERNISIERUNGEN AUCH GLEICHZEITIG FUNKTIONALE VERBESSERUNGEN IMPLEMENTIERT WERDEN?

***Funktionale Verbesserungen, die mit automatisierbaren Implementationsregeln beschrieben werden können, lassen sich mit geringem zusätzlichen Aufwand zusammen mit der Modernisierung durchführen. Dies bringt sogar Vorteile mit sich.***

Kann z.B. SOA im Verlauf eines Plattformwechsels eingeführt werden? Kann gleichzeitig für die Unicode-Unterstützung gesorgt werden?

Technisch gibt es keinen Unterschied zwischen erforderlichen Änderungen, um Anwendungen für eine neue Plattform zu transformieren, und funktionalen oder technischen Verbesserungen.

Generell kann jede Modernisierung oder Erweiterung, für die automatisierbare Implementierungsregeln definiert werden können, mit geringem zusätzlichem Aufwand zusammen mit einer Plattform-Migration durchgeführt werden.

Da die Fertigungsstraße existiert, die Knowledge Base aufgebaut ist, die erforderlichen Test-Szenarien durchlaufen werden, sind wesentliche Aufwände für eine zusätzliche Modernisierungsaufgabe schon erbracht.

SOA kann z.B. direkt durch den AMELIO Transformationsprozess oder später mittels Deltas SCORE-Produkten eingeführt werden. Die für eine Unicode-Unterstützung notwendigen Datenfluss-Analysen gehören zum Leistungsumfang von AMELIO, so dass sich solche funktionalen Verbesserungen leicht zusätzlich zu anderen Modernisierungsaufgaben durchführen lassen.

# KANN DER BUSINESS-WERT DER ANWENDUNGEN BEI EINER MIGRATION OHNE VERLÄNGERUNG DER PROJEKTLAUFZEIT ERHÖHT WERDEN?

***Eine n-tier-Architektur kann eingeführt werden.***

***Die Datenzugriffe können durch Nutzung von Daten-Services ersetzt werden, die dann auch anderen Anwendungen zur Verfügung stehen.***

***Browser-basierte Frontends können eingeführt werden.***

***Funktionen der Legacy-Anwendungen können als Business Services bereitgestellt werden und damit in anderen Anwendungskontexten verwendet werden.***

Aufgrund der tiefgehenden AMELIO-Analyse aller Source-Module stehen umfassende Informationen über die Anwendungen bereit.

Diese können genutzt werden, um Daten-Services (in COBOL oder C#/Java) anstelle der in die monolithischen Programme eingebetteten Datenzugriffe zu erzeugen. Diese Services werden genutzt von den migrierten Anwendungen und sie können genauso gut von C#- oder Java-Programmen genutzt werden, da sie sich ihnen wie normale C#- bzw. Java-Services präsentieren.

Solch eine Architektur-Modernisierung erlaubt die gemeinsame Nutzung der Daten-Services durch verschiedene Anwendungen und sie erleichtert Plattform-übergreifende Datenzugriffe.

Eine weitere Option ist das „Service Enablement“ der Anwendungen. Dabei werden Programmfunktionen als echte Business Services bereitgestellt, die eine echte C#- oder Java-Sicht auf die COBOL-Funktionen erlauben. Hierbei wird auf Deltas SCORE Adaptive Bridges zurück gegriffen.

# WAS SIND WESENTLICHE THEMEN EINER MIGRATION?

***Bei einem Migrationsprojekt stehen die Unterschiede zwischen alter und neuer Plattform im Vordergrund. Davon sind betroffen die DB-Systeme, TP-Monitore, Maskenformate, Sprachspezifika und systemtechnische Aufgaben, die auch auf der neuen Plattform bereitgestellt werden müssen.***

***Ein Plattformwechsel betrifft aber nicht nur die Anwendungen. Diese weitergehenden Aspekte müssen auch entsprechend berücksichtigt werden.***

Üblicherweise gibt es zahlreiche Unterschiede zwischen der alten und der ausgewählten neuen Plattform. Diese müssen systematisch ermittelt und behandelt werden.

Die gefundenen Unterschiede zwischen den beiden Plattformen hängen im Detail ab von den Plattformen selbst und den tatsächlich eingesetzten Systemspezifika.

Migrationshemmnisse der Plattformen können sein

- DBMS Call-Schnittstellen
- Erweiterungen der SQL-Standards (und natürlich auch Standards anderer Datenbankarten)
- spezifische Screen-Funktionalitäten und Screen-Handling
- spezifische Behandlung numerischer Felder
- spezifische Daten-Darstellungen aufgrund ihrer 9-Byte-Architektur
- Erweiterungen der COBOL-Sprache
- ….

Die neue Plattform muss auch bei anderen Themen aus dem Gesamtumfeld der Anwendungen berücksichtigt werden. So sind Plattformabhängigkeiten zu finden z.B. bei

- Backup- und Recovery-Prozeduren
- Release- und Patch-Verwaltung
- Interfaces und Connectivity
- Identitäts- und Zugriffs-Verwaltung
- Logging und Auditing
- Security-Handling
- Incident- und Change-Management

# WAS HAT DER BEGRIFF "CLEAN ROOM-KONZEPT" MIT DEM MIGRATIONSANSATZ VON AMELIO ZU TUN?

***„Clean Room“-Produktionen laufen vollständig von der Außenwelt abgeschottet ab. Jede Art von Verschmutzung, aber auch nicht dokumentierte Prozesse und Eingriffe in die Produktion sind ausgeschlossen.***

***Dieselben Prinzipien wenden wir mit AMELIO Modernization Platform auch auf Modernisierungsprojekte an.***

Mit AMELIO Modernization Platform geschaffene Migrationslösungen orientieren sich konsequent an den Gegebenheiten und Anforderungen der Praxis.

Um höchsten Qualitätsanforderungen und dem Anspruch von 100% Automation gerecht zu werden, überlässt die AMELIO-Fertigungsstraße während der gesamten Abfolge der Modernisierungsprozesse nichts dem Zufall oder der Geschicklichkeit des Bedieners.

Es werden nicht irgendwelche Teilschritte mit Einzelwerkzeugen von Hand angestoßen. Eine vollständige, sauber abgeschlossene Governance/Verwaltung aller Module ist fester Bestandteil des AMELIO-Ansatzes.

Von der Konsistenzprüfung bei der Order Entry (dem Eintritt in die Fertigungsstraße) bis zur abschließenden Auslieferung der transformierten Source-Module laufen alle Prozesse voll automatisiert ab:

-- Übergänge in Folgeprozesse sind nur möglich, wenn alle zu bearbeitenden Module eines behandelten Pakets die erforderlichen Eigenschaften aufweisen,

-- der Status eines Moduls wird nur dann in den festgelegten Folgestatus überführt, wenn alle vordefinierten Eigenschaften erfüllt sind.

Alle manuellen Eingriffe werden verhindert.

Wegen der konsequenten Ausschaltung aller äußeren Einflüsse nennen wir das mit AMELIO verfolgte Konzept "Clean Room-Konzept“.

# WIE UNTERSCHEIDEN SICH EMULATIONSLÖSUNGEN VON EINER ZU 100% AUTOMATISIERTEN MIGRATION?

***Jede Art von Emulation hält an alter Technologie fest und bringt diese alte Technologie auf die neue Plattform.***
***Diese Art von Migration endet letztendlich immer in n+1 technischen Parametern, fügt den bereits vorhandenen n technischen Parametern also mindestens einen weiteren neuen hinzu.***

Emulationen können als Lösungen sinnvoll sein für Anwendungen mit beschränkter Restlebensdauer, d.h. wenn die alte Funktionalität unverändert weiter genutzt werden soll, wenn also für diese Anwendungen keine Weiterentwicklung oder Wartung zu erwarten ist oder aber sogar ihre kurzfristige Ablösung geplant ist.

Ansonsten sehen wir Hardware- oder Software-Emulationen nicht als geeignete Lösungen für Plattform-Migrationen an, da sie alte Technologien konservieren und diese auf die neue Plattform übertragen. Die migrierten Anwendungen bleiben dabei abhängig von den Beschränkungen des alten Legacy-Systems, sie werden daran gehindert, die Möglichkeiten der neuen Plattform zu nutzen, das gesamte neue Potenzial wird jetzt und auch zukünftig nicht genutzt.

Ein häufiger Grund für einen Plattformwechsel ist das abnehmende Know-how bzgl. der alten Plattform. Für den Unterhalt der migrierten Anwendungen wird bei Emulationslösungen dieses zunehmend schwerer zu beschaffende Know-how dauerhaft benötigt.

Emulierende Komponenten sind typische „Footprint“-Lösungen, da die auf die neue Plattform verlagerten Anwendungen dort nur zusammen mit der Emulations-Umgebung genutzt werden können.

Der Anbieter der Legacy-Umgebung wird bei einer Emulationslösung ausgetauscht durch den Emulationsanbieter, was zu einer dauerhaften Herstellerabhängigkeit für die neue Plattform führt.

# WIE LASSEN SICH DIE TESTAUFWÄNDE BEI EINER ANWENDUNGSMIGRATION ODER -MODERNISIERUNG MINIMIEREN?

***Voraussetzung für eine nennenswerte Reduktion der Testaufwände ist die völlige Automatisierung der Transformationsarbeiten.***
***Eine 100%-ige Automatisierung ist der entscheidende Faktor für eine radikale Reduktion der Testaufwände mit Hilfe unseres Meta-Level-Testens.***

Entscheidende Projektfaktoren bei Massenänderungen sind die Tests. Die Testaufwände müssen reduziert werden, um nicht Zeit- und Kostenrahmen zu sprengen. Das Testkonzept eines Plattformwechsels muss genau definiert werden: nicht nur der geänderte Code ist zu testen, sondern sämtliche Plattformparameter. Tests müssen sich auf den Nachweis der Korrektheit konzentrieren, genauso aber auch auf nicht-funktionale Eigenschaften wie Verlässlichkeit, Skalierbarkeit, Sicherheit, Connectivity und Performance.

Unsere Strategie für das Testen der Code-Änderungen basiert auf dem Meta-Level-Testing und als Vorbedingung auf 100%-ig automatisierte Durchführung aller Änderungen.

Da Anwendungsfunktionen nicht von den Änderungen angerührt werden, ist es ausreichend, die Transformationsergebnisse mit einer ausgewählten Anzahl von Programmen zu verifizieren und nicht die transformierten Programme selbst.

So können die Testaufwände dramatisch reduziert und in die nicht-blockierende Vor-Produktionsphase verschoben werden.

In der Produktionsphase ist dann für die transformierten Programme nur noch ein Roll-out-Test erforderlich.

Zu Beginn der Tests für die neue Plattform steht ein Test der definierten Architektur. Schwerpunkt wird die Integration der Funktionalitäten der verschiedenen auf der neuen Plattform zu findenden Komponenten sein.

Die zweite Phase des Plattformtests wird sich um Verlässlichkeit, Skalierbarkeit, Performance und Stabilität der Plattform kümmern.

Der dritte Schwerpunkt der Teststrategie liegt auf Prozessen und Verfahrensabläufen. Für die Migration müssen alle Prozesse und Verfahren im Umfeld des Systems für das neue System geändert oder neu definiert werden. Tests der neuen Prozesse und Verfahren müssen sicherstellen, dass das neue System genauso gut gemanagt werden kann wie das alte. Dies gilt sowohl für den operativen Teil als auch für den Entwicklungsbereich des Unternehmens.

# WIE HOCH IST DIE TESTABDECKUNG DES MIT AMELIO MIGRIERTEN CODES?

***Die AMELIO-Methodik basiert auf dem 100%-igen Test aller Änderungen. Dabei wird Meta-Level-Testing verwendet, das die 100%-ige Testabdeckung unterstützt.***

100% aller von AMELIO Modernization Platform durchzuführenden Änderungen werden mittels Meta-Level-Testing getestet.

Hierbei werden nicht 100% aller Module getestet. Nur ein minimales, optimiertes Test-Set muss stattdessen getestet werden, das alle Transformationsregeln für alle Modernisierungsaufgaben abdeckt.

Dieser Ansatz konzentriert sich also auf das Testen der Transformationsalgorithmen und -regeln, anstatt alle transformierten Komponenten zu testen.

# WORAUS BESTEHT “META-LEVEL-TESTING”?

***Beim Meta-Level-Testing werden die Transformationsalgorithmen und Regeln anstelle aller transformierten Komponenten getestet. Das führt zu einer außergewöhnlichen Reduktion der Testaufwände nach Massenänderungen.***

Die Reduktion der Testaufwände ist der effizienteste Ansatz, um Projektaufwand, -kosten und -dauer zu reduzieren. Darauf zielt AMELIOs Meta-Level-Testing.

Die Qualität durchgeführter Massenänderungen, wie sie z.B. für Plattformwechsel, anwendungsübergreifende Modernisierungen oder Verbesserungen notwendig sind, ist ein sehr kritischer Faktor, falls diese Änderungen von Menschenhand durchgeführt werden und daher der „menschliche Faktor“ zum Tragen kommt. Nur 100%-ige Automation ist in der Lage, diesen Faktor auszuschließen.

Werden z.B. 90% der Änderungen automatisiert, verbleiben „nur“ (oder „noch“) 10% manuelle Änderungen. Das bedeutet in der Regel, dass 10% aller Änderungen über alle Programme hinweg manuell durchgeführt werden – und nicht, dass nur die Änderungen in 10% der Programme von Hand durchgeführt werden müssen.

Dies erfordert offensichtlich intensives Testen – nicht nur für 10% der Programme sondern für alle!

Eine 100%-ige Automation führt also direkt zu einer signifikanten Reduktion der Testnotwendigkeiten und Testaufwände.

Automaten sind sicherlich nicht immer und auch nicht unbedingt von vorneherein perfekt. Aber sie erledigen ihre Arbeit konsequent: Auch Fehler werden konsequent immer wiederholt. Andererseits: Wenn sie ihre Arbeit korrekt erledigen, erledigen sie diese immer korrekt. Daher reicht es, das Ergebnis eines Automaten (also das Ergebnis eines implementierten Algorithmus) ein einziges Mal mit nur einer kleinen Anzahl von Testfällen zu prüfen: Tests der Modifikationsergebnisse anstatt aller modifizierten Programme.

AMELIO Modernization Platform liefert eine Fertigungsstraße, in der die menschlichen Eingriffe auf ein Minimum reduziert sind. Die Source Code-Änderungen sind zu 100% automatisiert, menschliche Einmischungen sind nicht möglich. Nur diese Maßnahmen ermöglichten in einem zurückliegenden Projekt eine *Defect Injection Rate* von nur 0,00004% bzw. Fehlerquote von 0,004% bei 1,3 Millionen durchgeführten Änderungen.

Wir bezeichnen die AMELIO-Teststrategie als Meta-Level-Testing. Es wird vor den produktiven Cluster-Transformationen durchgeführt. Sein Zweck ist es, die Transformation selbst zu testen und nicht die transformierten Programme.

Schritte des Meta-Level-Testing s:

i) Coverage Analysis
Sie wird als erstes ausgeführt, damit alle relevanten Fälle getestet werden können. Nachdem alle Komponenten importiert wurden, wird dazu die aufgebaute Knowledge Base analysiert.
Ziel der Coverage Analysis ist es, die Programme für ein minimales Test-Set zu ermitteln, die sämtliche verschiedenen Transformationsfälle abdecken.

ii) Die Objekte dieses Test-Sets werden transformiert und die Transformationsresultate anschließend überprüft.
Dieser Vergleich der Transformationsergebnisse mit den gewünschten Ergebnissen kann bedarfsweise durch einen maschinellen Code-Regression-Test erfolgen.

iii) Compiles der repräsentativen Code-Module verifizieren die syntaktische Korrektheit.

iv) Zum Schluss des Meta-Level-Testing s erfolgt der Runtime-Test der ausgewählten transformierten Programme.

Die Gesamt-Testaufwände werden so dramatisch reduziert und obendrein auch noch nach vorne in die unkritische „Non-Blocking-Phase“ vor der produktiven Transformation verlagert.

# GEFÄHRDEN MIGRATIONS- ODER MODERNISIERUNGSPROJEKTE ZWANGSLÄUFIG DIE STABILITÄT DER ANWENDUNGEN?

***Unerwartete Side-Effects auf die Geschäftslogik werden durch die 100%-ige Automatisierung der Änderungen ausgeschlossen, so dass bei Einsatz von AMELIO Modernization Platform eine Gefährdung der Stabilität der geschäftskritischen Anwendungen ausgeschlossen werden kann.***

Ein Migrations-/Modernisierungsprojekt muss selbstverständlich sicherstellen, dass sich Funktionalität und Qualität der Anwendungen durch die Transformationen nicht ändern.

Bei manuellen Änderungen wächst die Gefährdung mit der Anzahl der durchgeführten Änderungen. Im Falle 100% automatisierter Änderungen ist die Anzahl der durchgeführten Änderungen ohne Bedeutung, da garantiert ist, dass sich keine „zufälligen" Funktionsänderungen einschleichen können.

Natürlich gilt auch für automatisierte Änderungen das bekannte Prinzip: so wenig Änderungen wie möglich – aber so viele wie nötig (aber auch keine weniger!)

# SIND SPRACHTRANSFORMATIONEN MÖGLICH UND SINNVOLL?

***Auch Sprachtransformationen stellen für AMELIO kein Problem dar.***
***In Abhängigkeit von Quell- und Zielsprache stellt sich aber die Frage nach der Zweckmäßigkeit einer Sprachtransformation.***

Sprachtransformationen stellen kein technisches Problem dar. Es gibt Produkte, die „mehr oder weniger" automatische Sprachtransformationen z.B. von COBOL nach C# oder Java anbieten. So etwas wäre auch mit der AMELIO Modernization Platform möglich.

Sprachtransformationen sollten die zugrundeliegenden Paradigmen berücksichtigen. Wir halten deshalb die direkte Transformation einer prozeduralen Sprache wie z.B. COBOL in eine OO-Sprache wie C# oder Java nicht für befriedigend. Das Ergebnis solch einer Überführung wird weder die C#-/Java-Programmierer zufriedenstellen noch die COBOL-Programmierer.

Für Anwendungen mit nur noch einer kurzen verbleibenden Lebenserwartung und keiner erwarteten Weiterentwicklung oder Wartung mag solch eine Sprachtransformation ausreichen. Die Erfahrungen zeigen jedoch, dass der resultierende Java-/C#-Code kein „echter" Java-/C#-Code ist. Das Ergebnis wird eine Art „JOBOL" sein, wartbar weder für die COBOL- noch für die Java-/C#-Programmierer.

Andere Sprachtransformationen wie z.B. Delphi oder C++ nach C#/Java oder die Ersetzung von Frameworks oder Klassenbibliotheken eignen sich aber sehr gut für automatische Transformationen.

Ist die stärkere Nutzung von Java oder C# in dem Unternehmen der Anlass für eine ins Auge gefasste Sprachtransformation, dann besteht eine bessere Lösung möglicherweise darin, die Business-Logik der COBOL-Anwendungen den Java-/C#-Programmen auf eine Java-/C#-typische Art als Services zur Verfügung zu stellen. Die Wiederverwendung von Legacy-Funktionalität in einer modernen SOA-Welt geht über eine reine 1:1-Migration hinaus. Dafür steht das Delta-Produkt SCORE Adaptive Bridges bereit.

# WELCHE BETEILIGUNG DES KUNDEN WIRD IN EINEM AMELIO-MODERNISIERUNGSPROJEKT BENÖTIGT?

***Wissen und Erfahrung des Kunden werden hauptsächlich für die Definition der Modifikationsanforderungen und zu einem geringeren Teil für die stark reduzierten Tests benötigt.***

*i)* *Definition des Projektumfangs:*
In dieser Phase legen Kunde und Delta gemeinsam die Transformationsregeln fest. Für die Festlegung der zu implementierenden Modernisierungswünsche ist der Kunde verantwortlich.

ii) *Code-Import:*
Hierbei kann Unterstützung des Kunden bei auftretenden Fragen z.B. zu speziellen Programmierstilen erforderlich werden.

iii) *Coverage Analysis:*
Nur minimale Kundenunterstützung erforderlich.

iv) *Definition des neuen Frameworks (also der Standards und der die Standards unterstützenden Komponenten) und abschließendes Design der Transformationsregeln:*
Diese Aufgabe wird von Delta in Zusammenarbeit mit dem Kunden durchgeführt.

*v)* *AMELIO-Konfiguration:*
Wird von Delta erledigt, keine Kundenunterstützung erforderlich.

*vi)* Meta-Level-Testing:
Mit Kundenunterstützung

*vii)* *Datenmigrations-Programme*
Können von Delta auf Basis der beim Code-Import gesammelten Informationen generiert werden.

*viii)* *Betrieb der AMELIO-Fertigungsstraße:*
Durch Projektmitarbeiter des Kunden oder bedarfsweise auch durch Delta

*ix)* *Roll-out-Tests der transformierten Anwendungs-Cluster und Inbetriebnahme:*
Durch den Kunden

Am Anfang eines jeden Projekts führen wir eine Vorstudie der verfügbaren Source-Module durch. Dies ist dann die Basis für die Diskussion der Migrations-/Transformations-Aufgaben von Schritt (i).

Die Ergebnisse dieser Vorstudie sind Basis für Planung und Schätzungen, auf denen die folgenden Schritte (ii) und (iii) basieren. Eine Spannbreite der „von – bis"-Kosten des gesamten Projekts wird bestimmt.

Nachdem (ii) und (iii) ausgeführt sind und die Ergebnisse „Framework-Definitionen" des Schritts (iv) vorliegen, können die Schritte (v) – (ix) geplant und kalkuliert werden.

# WIRD DURCH AMELIO EINE DER ZIELPLATTFORMEN MICROSOFT WINDOWS ODER UNIX/LINUX FAVORISIERT?

***AMELIO unterstützt nahezu alle Plattformen (Mainframe, UNIX/Linux, Microsoft Windows).***
***Für eine 1:1-Migration sind nach unserer Einschätzung Microsoft Windows und UNIX/ Linux gleich gut geeignet. Beide werden von AMELIO als Zielplattform voll unterstützt.***

AMELIO unterstützt eine Vielzahl von Plattformen. Wir bewerten Microsoft Windows und UNIX/Linux als technisch gleichermaßen geeignet für eine 1:1-Plattformmigration und wir unterstützen beide. Beide haben sowohl Vor- als auch Nachteile.

Weitere UNIX-Plattformen wie Oracle Solaris, IBM AIX und HP HP-UX werden selbstverständlich auch unterstützt.

Für die Zielplattformen Windows und UNIX/Linux werden verschiedene Konstellationen incl. Web Forms, .NET, Internet Information Services (IIS) für Microsoft Windows oder JSP oder JSF für UNIX/Linux unterstützt.

Der Hauptunterschied zwischen Microsoft Windows und UNIX/Linux könnte zum Zeitpunkt nach der Migration bedeutsam werden.

Die Microsoft Windows .NET-Server-Plattformen sind nicht nur gut gerüstet für 1:1-Migrationsansätze, sie ermöglichen auch perfekt alternative Nutzungen der modernisierten Anwendungen. Mit dem.NET Framework 4.0, der Windows Communication Foundation (WCF) und der überarbeiteten Windows Workflow Foundation (WWF) stehen gute Tools für das Management sowohl Service-orientierter (Web-)Anwendungen als auch Enterprise-Anwendungen bereit.

Für den Plattform-Support ist mehr Personal mit technischem Know-how im Microsoft Windows-Umfeld als für UNIX/Linux verfügbar.

# WELCHE PLATTFORMEN UNTERSTÜTZT AMELIO?

AMELIO ist für alte und neue Architekturen, Plattformen, Sprachen, Frameworks und Technologien einsetzbar und wird genau auf die Projektbedürfnisse und –aufgabenstellung zugeschnitten.

**Sprachen**

AMELIO analysiert stets den kompletten Source-Code aller Anwendungskomponenten inklusive aller Meta-Level-Komponenten wie Macros, Copybooks etc.

- COBOL
  inklusive aller Dialekte (z.B. Unisys ACOB/UCOB, IBM OS/VS COBOL, DOS/VS COBOL, COBOL II, COBOL 370, Micro Focus, AcuCOBOL, Ryan-McFarland, Liant, Fujitsu, Microsoft)
- PL/I
- Delta/ADS, Delta PL/I
- Embedded SQL, Dynamic SQL, SQL Call Interface

**Betriebssysteme**

- Bull DPS6 GCOS6, DPS7 GOCS7, DPS8 GCOS8
- Fujitsu-Siemens BS2000/OSD
- HP HP-UX, Tandem NonStop Guardian, Tandem NonStop OSS
  DEC VAX VMS, OpenVMS
- IBM AIX, DOS/VSE, z/OS
- ICL VME, VME/B, VME/K, VME-X
- Microsoft Windows 95, 98, NT, 2000, XP, 2003, Vista, Windows 7
- Sun/Oracle Solaris
- Unisys OS 2200
- UNIX, Linux – verschiedene Hersteller

**DBMS**

- Sequentielle Dateien (Flat Files) und Index-sequentielle Dateien für alle Systeme
- Bull IDS2, INTEREL/RFM
- Fujitsu LEASY, SESAM, SQL Server, UDS
- HP Tandem NonStop SQL/MX, SQL/MP, DEC Rdb
- IBM DB2, IBM DL/I, IMS/DB, Informix

- ICL IDMS, IDMSX
- Ingres
- Microsoft SQL Server
- Oracle
- Software AG Adabas
- Sybase SQL Server
- Unisys RDMS, RSA RDMS

# AMELIO MODERNIZATION PLATFORM

***AMELIO® Modernization Platform ist ein maßgeschneidertes Entwicklungswerkzeug, das die Transformation großer IT-Anwendungen vollständig automatisiert:***
***100% automatisch und deshalb sicher, zuverlässig und fehlerfrei.***

AMELIO Modernization Platform

- analysiert und ändert Anwendungen mit Hilfe eines voll automatisierten, regelbasierten Prozesses, der exakt auf die jeweiligen Anforderungen zugeschnitten ist,
- reduziert Testaufwände drastisch,
- implementiert Änderungen, ohne andere Projekte oder laufende Maintenance zu blockieren,
- garantiert zu jeder Zeit die Stabilität und Integrität der Anwendungen,
- dokumentiert alle Änderungen revisionsgerecht,
- lässt die Wahl zwischen schrittweiser Umsetzung und „Big Bang".

## EINSATZBEREICHE FÜR AMELIO MODERNIZATION PLATFORM:

- **Massenänderungen:**
  Änderung von Datenformaten, Kunden-, Konto- oder Versicherungsnummern, Euro, Swift, UTF-16 (Unicode) für EU-Harmonisierung, …
- **Anwendungsmodernisierung:**
  Plattformwechsel, Framework-Anpassung / Austausch, …
- **Architekturtransformation:**
  Service Enablement, Modularisierung, Plattformneutralisierung
- **Sprachtransformationen:**
  (4GL, Delphi, C++ etc.)
- **Quality Assurance**

Unsere Kunden bestätigen: Große Änderungsprojekte können auf diese Art in kürzerer Zeit, mit äußerster Sicherheit, mit weniger Ressourcen und mit allerhöchster Qualität durchgeführt werden – durch vollständige Automation.

## WWW.D-S-T-G.COM/AMELIO



Bestellnummer: MT 11'058.02 – September 2011