



On the Fly Transformation Into CA Gen

Mustafa Arikan, Arikan Productivity Group GesmbH
mustafa.arikan@arikan.at

Session Track 3

12th October 11:30 12:10

Biography

Mustafa Arikan studied industrial engineering, mathematics and computer science in Istanbul and in Vienna and finished his education in 1986. He has meanwhile 29 years industrial experience in IT and operations research. He worked for vendors like IBM and as technology partner of Computer Associates for various large scale companies and won many IT awards throughout his career so far. His companies serve in Austria and Turkey and in cooperation with partners in over 10 countries mainly in software modernization.



Agenda

Software Modernization.

Legacy Code

Program Transformation.

Goal of Legacy Transformation.

Transformation.

Modernization state-of-the-art

Metamodel Based Transformation

Documentation

Demo.

Software Modernization

Legacy Transformation, or legacy modernization, refers to the rewriting or [porting](#) of a [legacy system](#) to a modern [computer programming](#) language, software libraries, protocols, or hardware platform. Sometimes referred to as software migration, legacy transformation aims to retain and extend the value of the legacy investment through migration to new platforms.

Some parts of this presentation are taken from WIKIPEDIA.

www.wikipedia.org

Software Modernization – Legacy Code

A [legacy code](#) is any application based on older technologies and hardware, such as mainframes, that continues to provide core services to an organization. Legacy applications are frequently large and difficult to modify, and scrapping or replacing them often means re-engineering an organization's business processes as well. However, more and more applications that were written in so called modern languages like java are becoming legacy. Whereas 'legacy' languages such as Cobol are top on the list for what would be considered legacy, newer languages can be just as monolithic, hard to modify, and thus, be candidates of modernization projects

Modernization – Program Transformation

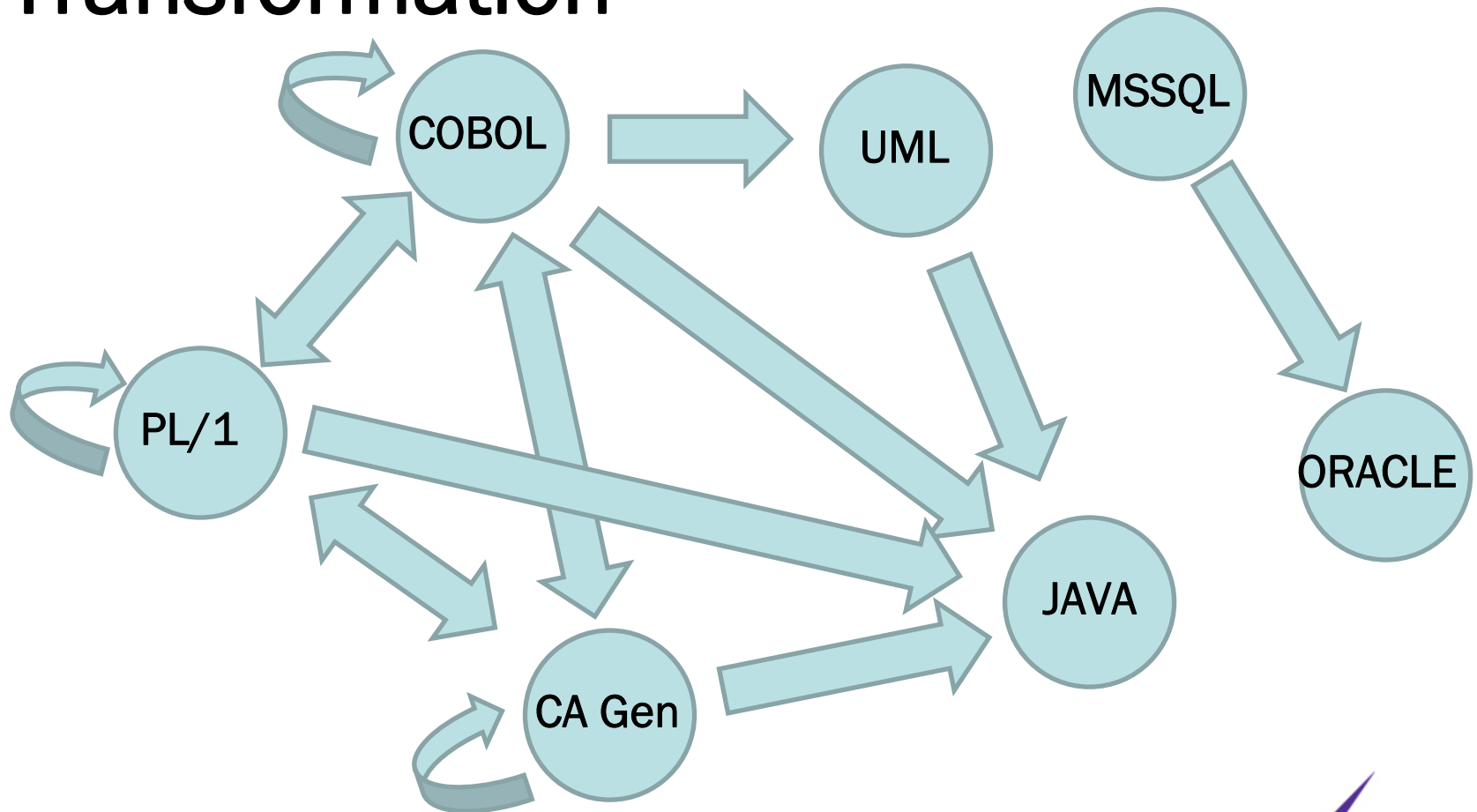
Re-implementing applications on new platforms in this way can reduce operational costs, and the additional capabilities of new technologies can provide access to functions such as web services and integrated development environments. Once transformation is complete and functional equivalence has been reached the applications can be aligned more closely to current and future business needs through the addition of new functionality to the transformed application. The recent development of new technologies such as [program transformation](#) by software modernization enterprises have made the legacy transformation process a cost-effective and accurate way to preserve legacy investments and thereby avoid the costs and business impact of migration to entirely new software.

Goal of Legacy Transformation

The goal of legacy transformation is to retain the value of the legacy asset on the new [platform](#). In practice this transformation can take several forms. For example, it might involve translation of the source code, or some level of re-use of existing code plus a Web-to-host capability to provide the customer access required by the business. If a [rewrite](#) is necessary, then the existing business rules can be extracted to form part of the statement of requirements for a rewrite.

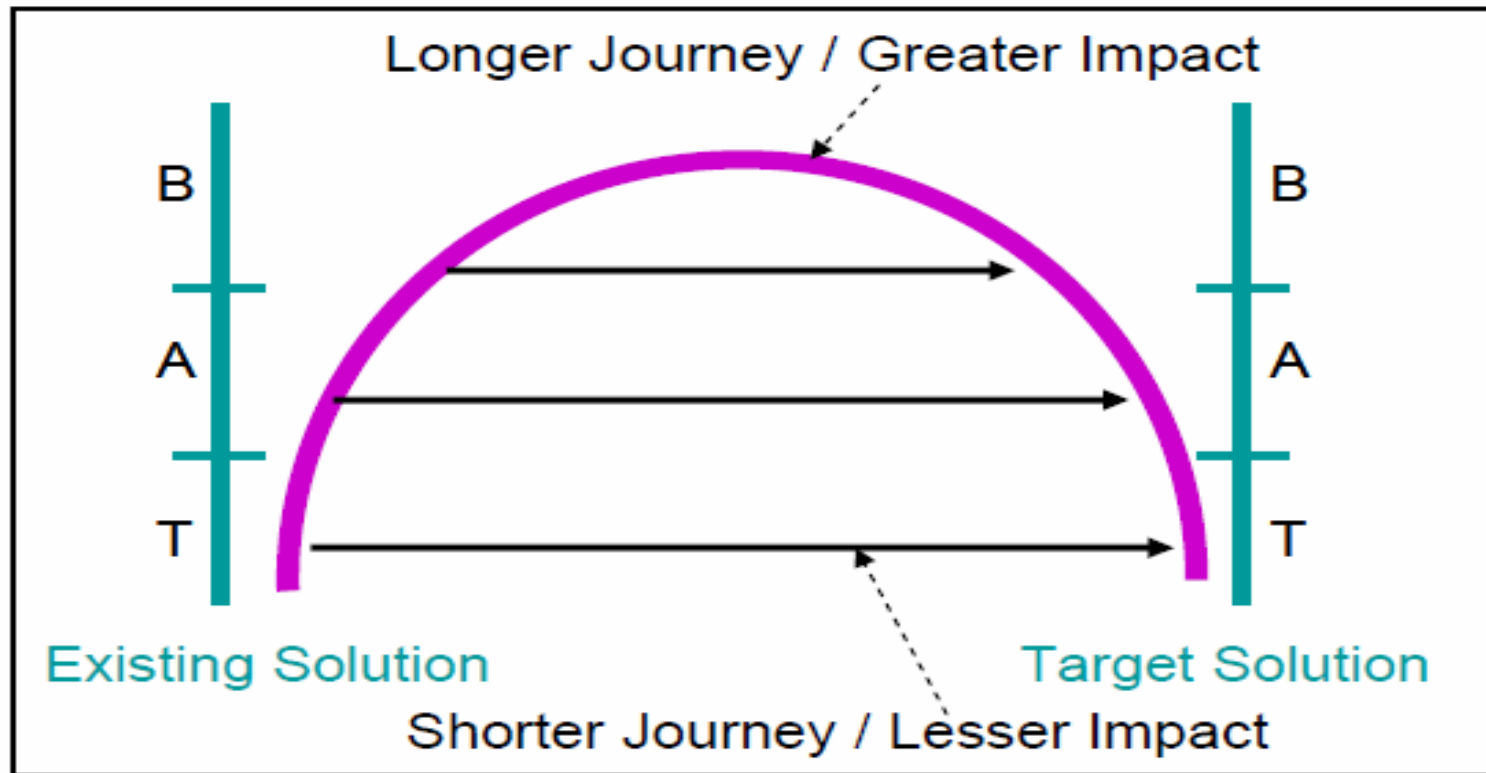
When a software migration reaches functional equivalence, the migrated application can be aligned more closely to current and future business needs through the addition of new functionality to the transformed application.

Transformation



Transformation

<http://www.omg.org/docs/admtf/07-12-01.pdf>



Metamodel

Metamodeling, or *meta-modeling* in [software engineering](#) and [systems engineering](#) among other disciplines, is the analysis, construction and development of the frames, rules, constraints, models and theories applicable and useful for [modeling](#) a predefined class of problems. As its name implies, this concept applies the notions of [meta-](#) and modeling.

Formal Grammar

A formal language is a set of words, i.e. finite strings of letters, symbols, or tokens. The set from which these letters are taken is called the alphabet over which the language is defined. A formal language is often defined by means of a formal grammar (also called its formation rules); accordingly, words that belong to a formal language are sometimes called *well-formed words* (or well-formed formulas).

A formal grammar (sometimes simply called a grammar) is a set of rules for forming strings in a formal language. These rules that make up the grammar describe how to form strings from the language's alphabet that are valid according to the language's syntax. A grammar does not describe the meaning of the strings—only their location and the ways that they can be manipulated

Metamodel Creation from Tool MM

Java - gen.diag - Eclipse SDK

File Edit Navigate Search Project Run Window Help

Package Explorer Gen Schema *gen.diag Outline

204. Elementary Attribute --> 202. Attribute
205. Special system defined attribute(53) -->
206. User defined attribute(54) --> 204. Elen
207. Relationship Aggregation(115) --> 199.
208. Relationship Aggregation Mbrship(116) -
209. Data Box --> 199. Entity Model Element
210. Partitioning(75) --> 209. Data Box
211. Data Definition --> 209. Data Box
212. Subject Area(76) --> 211. Data Definitio

Select
Marquee
Reference
Two-way Reference
Inheritance
Note Link

UML
Package
Class
Enum
Sticky Note
Attribute
Operation
Enum Literal

212. Subject Area(76) --> 211. Data Definitio

Portable across environments?: char
Proxy Flag (Y/yp): char
SUBJ Type (Spec/Impl/General/sp): char
Component Class UUID: string
Component Client Type Lib UUID: string
Component Type Lib UUID: string
Interface UUID: string
Java proxy package name: string
Subject area char filler 2: char
Subject area char filler: char
Subject area string filler 1: string
Subject area string filler 2: string
consists of: Subject area usage
contains: Subject Area
detailed by: Highest Level Entity Type G
is described by: Relationship Aggregatio
is implemented by: Subject Area
is part of: Subject area usage
scopes: Art for a diagram
scopes typemap: Type Map

CONTNDBY
CNTNDBYS
FOUNDBY

213. Entity Type --> 211. Data Definition

SubjectArea

- phase: EChar
- description: EString
- name: EString

RelationshipMembership

- name: EString
- cardinality: EChar
- modifyingOrReferencing?: EChar
- cascaDe,Restrict,Nullify,Pendant: EChar
- expectedAverageCardinality: EInt
- expectedMaximumCardinality: EInt

HighestLvAnalysisEntityType

- averageNumberOfOccurrences: EInt
- maximumNumberOfOccurrences: EInt
- percentageOfGrowth: EInt
- name: EString
- dataStructureDiagramName: EString
- category: EChar
- description: EString

UserDefinedAttribute

- name: EString
- length: EInt
- description: EString
- dataStructureDiagramName: EString
- type(Basic,Derived,Designed): EChar
- domain: EChar
- optionality: EChar

1..* +contains 0..*
0..1 +containedBy 1..*
1 +detailOf 1..*
1 +describedBy 0..*
1 +describedBy 0..*

defaultPackage
SubjectArea
HighestLvAnalysisEntityType
UserDefinedAttribute
RelationshipMembership
Procedure
ImportViewSet
ExportViewSet
LocalViewSet
EntityActionViewSet
ProcedureStep
Screen
EntityViewDefinition
Create
Associate
DialogBox
WindowLiteralText
HighestLevelDesignerEntType

Problems Javadoc Declaration DumpFileView Description

Number	CMD Type	Element Type	Parameters	Schema Info	Status
Transaction 3					ignored
Transaction 4					ignored
Transaction 5					MetaElement genrated
4400	AO	Subject Area	obj-id = 984	212/76, SUBJ	MetaElement genrated
4401	MP	Phase	obj-id = 984, new value = N	256, PHASE	MetaElement genrated
4402	MP	Description	obj-id = 984, new value = subjectarea	79, DESC	MetaElement genrated
4403	MP	Description	obj-id = 984, new value = subjectarea	79, DESC	ignored
4404	MP	Name	obj-id = 984, new value = AT	224, NAME	MetaElement genrated
4405	AA	contains	src-obj-id = 87, dest-obj-id = 984	36, CONTAINS	
4406	AA	uses subiect areas	src-nh-id = 1, dest-nh-id = 984	306, USES SUBJ	

11M of 21M

Metamodel Generation from Grammar

The screenshot displays the ANTLR Workbench interface for the file `\\Maria01\\everyone\\LIN\\Gen_Parser\\Gen_View.g`.

Top Pane (Grammar): Shows the source grammar `Gen_View.g` with the following content:

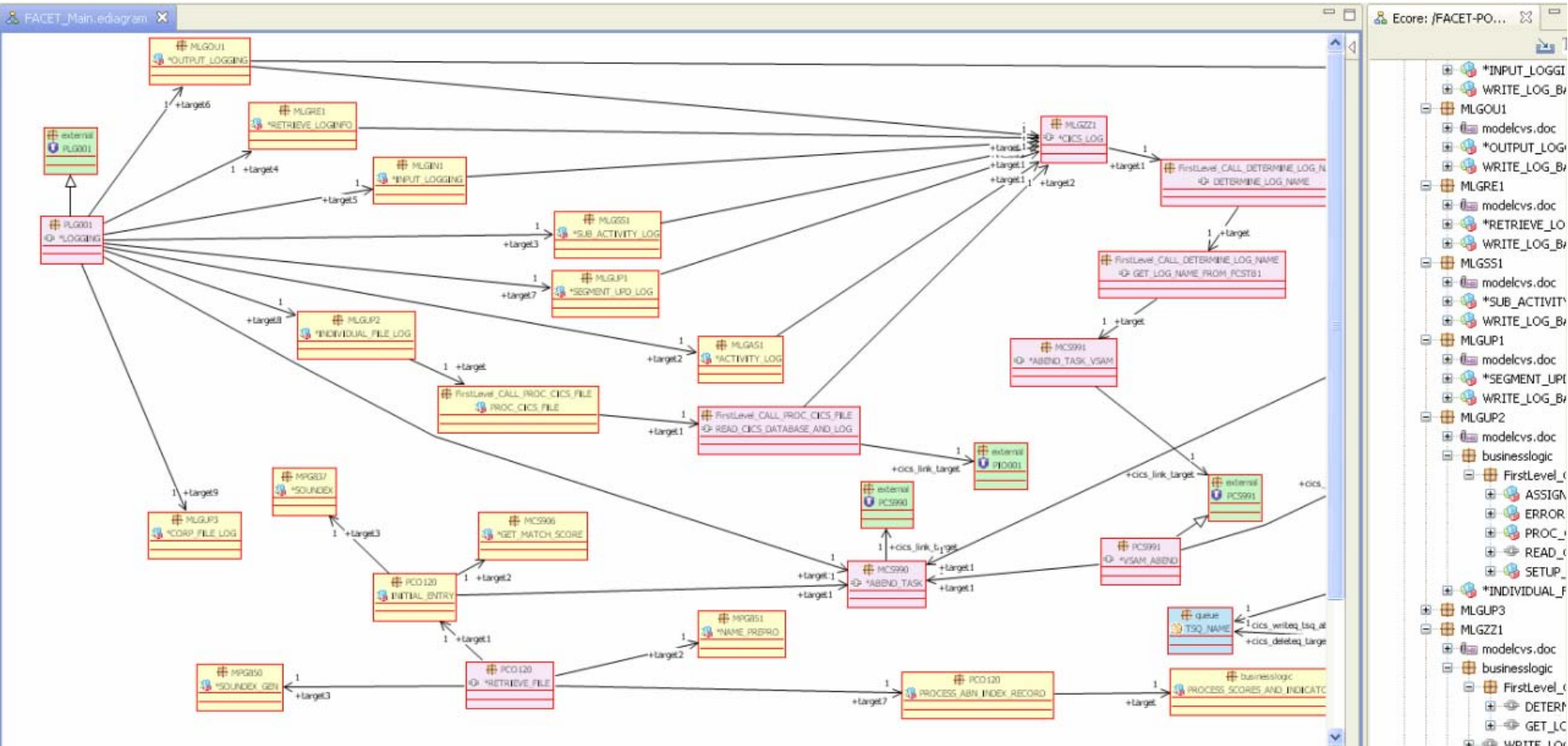
```
grammar Gen_View;  
  
options {  
    k = 4;  
    output=AST;  
}  
  
@header {  
    package at.arikan.cobol antlr;  
    import at.arikan.modelcvs.legacy antlr.parse.DummyActionHandler;  
    import at.arikan.modelcvs.legacy antlr.parse.IActionHandler;  
    import at.arikan.modelcvs.legacy antlr.parse.TokenType;  
    import java.util.ArrayList;  
    import java.util.LinkedHashMap;  
}  
  
@lexer::header {  
    package at.arikan.cobol antlr;  
}  
  
@rulecatch {  
}  
  
@members {  
    private IActionHandler actionHandler = new DummyActionHandler();  
  
    long lineNumber = 0;  
    protected void mismatch(IntStream input, int ttype, BitSet follow)  
        throws RecognitionException {  
        // ...  
    }  
}
```

Left Pane (Project Explorer): Lists the project structure for `Gen_View.g`, including `view`, `univName`, `common_name`, `view_definition`, `view_import`, `view_export`, `view_local`, `view_entityAction`, `view_entityView`, `view_attribute`, and terminal symbols `IDENT`, `INTID`, `DATELIT`, `INTLIT`, `DECLIT`, `STRING_LIT`, `DIGIT`, `LETTER`, and `WS`.

Bottom-Left Pane (Generated Code): Shows the generated metamodel code. The `IMPORTS:` section includes `Entity View imp person` and various attributes like `phone (mandatory)`, `email (mandatory)`, `jobdescription (mandatory)`, `department (mandatory)`, `surname (mandatory)`, `name (mandatory)`, and `number (mandatory)`. The `EXPORTS:` section includes `Entity View exp person`.

Bottom-Right Pane (Diagram): Displays a diagram of the metamodel elements and their relationships. The elements are `Entity`, `View`, `univName`, `univName`, `(`, `mandatory`, `,`, `transient`, `,`, `import`, and `only`. The `univName` elements are linked to `imp` and `person` respectively.

Documentation



DEMO



Q&A