

# Natural to Java Migration



## Highlights:

- Automated tools transform Natural and Adabas workloads in a timely, cost-effective manner
- Comprehensive service approach ensures that your complete environment is addressed
- Preserve as much or as little of your current look and feel as desired
- Extensive preparation and analysis phases mean no project surprises down the road

## Expand the potential of your legacy applications

Until now, moving Natural and Adabas application environments to mainstream technologies may have been too costly and time-consuming to consider and implementing packaged application alternatives meant sacrificing valuable business logic and custom functionality. Whether you are looking to preserve the look and feel of your current Natural Adabas applications or evolve your environment into a multi-tier Java application with a modern relational database over time, Clernity offers several options to enhance and extend the business value of your legacy workloads and can do so at the pace and budget level fitting to your business needs.

## A complete Natural to Java transformational process

Using automated migration tools and a comprehensive service methodology, Clernity offers an efficient, powerful transformation process to move terminal-based Natural Adabas environments to multi-tier Java applications integrated with modern relational databases.

Clerity's Natural-to-Java process includes the following stages:

- Discovery
- Application assessment
- Relational database (RDBMS) schema generation
- Source code conversion
- Data migration
- Performance tuning and customization
- Testing and production cutover

As depicted in Figure 1, *Natural-to-Java Process Overview*, with the Clernity approach Natural source code is analyzed and parsed in detail. The parsed output is then stored in an RDBMS based repository. All subsequent schema and data migration, as well as source code conversion, is then driven from the repository database.

The functionality of the resulting Java RDBMS application is identical to the original Natural Adabas application. The transformation strategy also preserves the look and feel of the Natural application while enabling continuing development in a modern Web-oriented environment. The strict functional equivalence approach with Natural to Java provides the additional benefit of simplified testing and training, getting your organization down the road with mainstream technologies in a fast, efficient manner

**Project discovery**

Before formally analyzing your Natural Adabas workload, Clerity will work with your organization in a Discovery phase to confirm all the business and technical goals of your project.

Conducted by a senior project manager, Clerity holds a series of meetings that include all of the parties involved in the project. During these meetings, the level of project management desired, as well as testing and production readiness criteria are established. In addition to your software development staff, we encourage members of your infrastructure support team to be involved as well. The project budget and timetable are set at this time after all inputs have been gathered. Throughout the process, Clerity performs stringent testing of migrated data between two environments to insure that desired results have been achieved and to mitigate migration risk.

**Application assessment**

*Getting started*

Clerity’s Natural to Java conversion process starts with collecting all the Natural source code from your application portfolio, including File Description Tables (FDTs) and Data Definition Modules (DDMs), in Systrans format. This can be

easily accomplished using the Natural Systrans Adarep utilities and special instructions we provide. As it is crucial that all source code be supplied, the focus of preliminary processing is to identify any missing modules.

The Systrans file is transmitted to Clerity’s migration center where the FDTs, DDMs, Maps and all other Natural source objects are parsed into Abstract Syntax Tree format.

Then the parsed data is extracted and loaded into the repository. For each Natural module there is a corresponding entry in the repository database with its associated statements, data elements, references, and all other information required to rebuild the application preserving all of its functionality.

*Analysis*

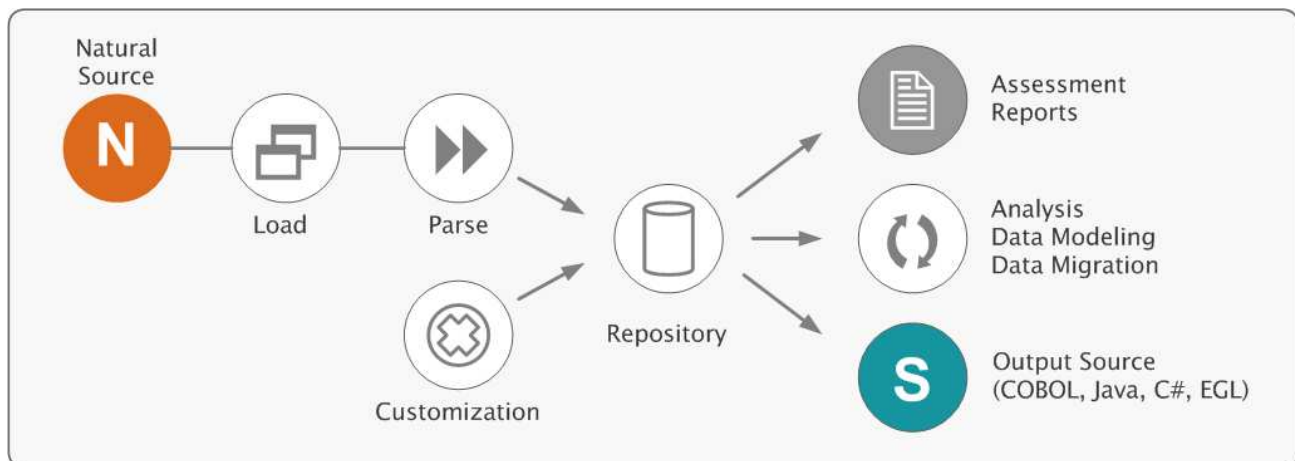
Clerity initially performs an automated analysis of your parsed data to determine the feasibility of conversion, and to identify problematic code such as dynamic code. From this analysis, detailed assessment reports are generated containing a comprehensive description of the source code and application details which aid in estimating the size of the migration. The repository database and reports can also support application mining for migration planning and the development of test scenarios.

Comprehensive inventory reports showing missing modules ensure the completeness and integrity of the code prior conversion. During the assessment, any modules or DDMs that are identified as unused will be purged from the SAR database before proceeding with subsequent processing stages.

**RDBMS Schema Generation**

Once the parsed data is loaded into the SAR, you then have the option of specifying field and table names that will be

Figure 1. Natural-to-Java Process Overview



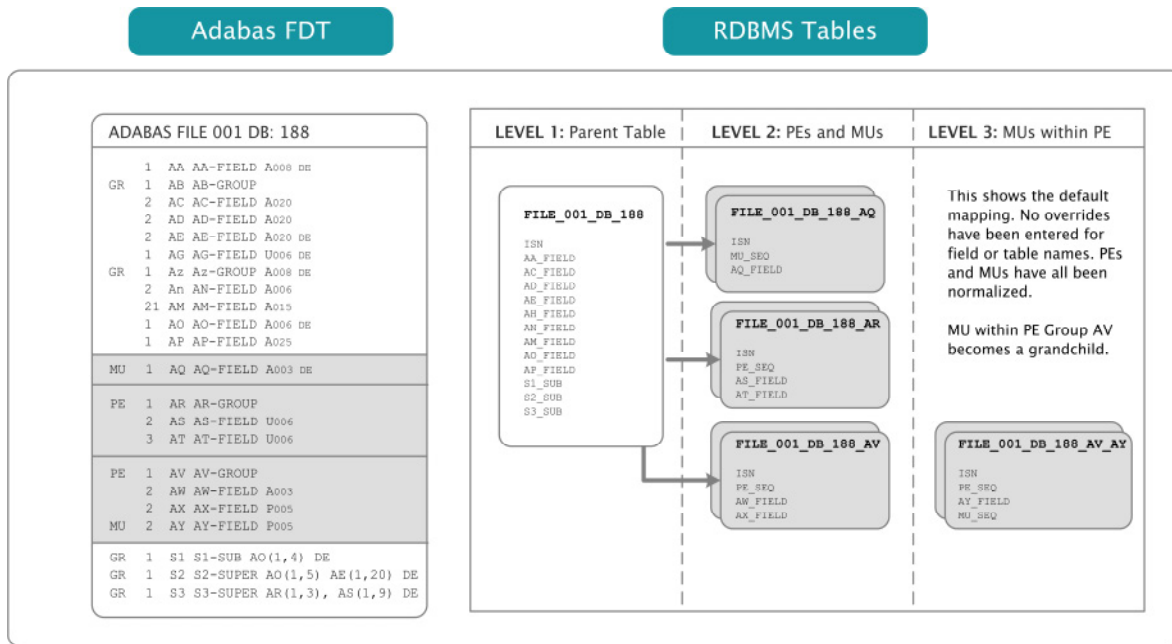


Figure 2. Sample Adabas to RDBMS Schema Conversion

used in the creation of the new schema, as well as some ability to change data types and sizes. The Natural to Java default field and table names for FDTs are hard to decipher, as they are built from the Adabas short names. However, you have the option of overwriting table and field names so that the new schema definition will be more meaningful to your application developers going forward. By default, relational tables will be built from the conversion details stored in the SAR, according to the following mapping rules:

- Each FDT is converted to at least one corresponding relational table in the target RDBMS whose unique key will be the ISN of the original Adabas record. This table will be comprised of all the fields from the FDT with the exception of Periodic Groups (PEs), Multiple Usage fields (MUs) and Group fields
- PEs are normalized, becoming Child Tables with Foreign Keys to the Parent Table
- Similarly, by default MUs are normalized, and Child Tables created with Foreign Keys to the Parent Table
- Optionally, an MU field may be mapped to a partitioned text field that contains all occurrences of the MU delimited by a special character, or a separate field might be created to hold each occurrence of the MU field. (e.g.: ADDRESS\_LINE1, ADDRESS\_LINE2, etc.)
- Each Superdescriptor (SP) and Subdescriptor (SB) is converted to a physical field in the corresponding table in the target database. A database trigger is created

which maintains the structure of the SP or SB components when these are modified in the database. Under certain circumstances, SQL can be used to copy the functionality of the SP/SB, eliminating the need for a physical field

- Phonetic descriptors will be replaced by database triggers that implement a Soundex algorithm
- Descriptors become indexes in the new schema

Schema generation is an iterative process involving your review and input until a satisfactory schema is agreed upon. Once your desired schema is obtained, the data from the SAR is used to generate the CREATE TABLE, PRIMARY KEY and FOREIGN KEY statements.

An example of a simple Adabase to RDBMS schema conversion is illustrated in Figure 2.

#### Source code conversion

Clarity's Natural to Java source code conversion strategy is based on three key components:

##### 1) Logic Base Class

The Logic Base Class is series of custom Java methods implementing the Java equivalent functionality of the Natural language syntax. In general, each Natural statement is mapped to a Java method on a one-to-one basis. The Natural to Java methods are designed to give the converted code a Natural look and feel.

2) Support Base Classes

Support Base Classes are a series of custom Java classes to provide the Java equivalent functionality for certain Natural runtime components and features. Some examples are:

- NatSession – maintains and controls the Natural to Java runtime environment
- NatSysVar – manages the system variables such as \*DATX and \*COUNT
- NatStackItem – manages items passed on the stack
- NatWorkFile – manages the disk I/O for the Natural Read/Write Workfile statements
- NatDDM – maps View structures to the RDBMS schema

3) Translation Engine

A SAR driven tool that converts each Natural module source code, on a statement-by-statement basis, to a functionally equivalent Java class using methods of the Logic base class

and native Java statements to create Java logic that will execute exactly as it was executed in the Natural environment. Each Natural source module is converted to a Java class that extends the Logic base class. The new Java classes corresponding to the Natural program.

Data migration

Clarity’s Natural to Java process also provides tool-supported data migration from Adabas production files to RDBMS tables. An automated process combines data from the SAR with the newly created RDBMS schema to enable the following:

- Creation of an RDBMS based data staging area derived from the transformed Adabas schema
- Population of the staging area with data from the Adabas files to facilitate subsequent data validation, data type mapping and data warehousing activities. The Adabas data will be extracted from data files produce using standard Adabas utilities

Figure 3. Natural to Java Example

Natural	Converted Java
<pre> 010 DEFINE DATA LOCAL 020 1 EMPLOYEES VIEW OF EMPLOYEES 030   2 PERSONNEL-ID 040   2 FULL NAME 050     3 FIRST NAME 060     3 NAME 070     3 MIDDLE-NAME 080   2 SEX 090   2 FULL-ADDRESS 100     3 ADDRESS-LINE (1:4) 110     3 CITY 120     3 POSTAL-CODE 130     3 COUNTRY 170 1 #NAME (A40) 0 180 END-DEFINE                     </pre> <p>Natural Look and Feel</p>	<pre> //Define Data Local(); View (1, "EMPLOYEES", new dEMPLOYEES()); Decl( 2, "PERSONNEL-ID", 'N',8); Decl( 2, "FULL NAME", 'G'); Decl( 3, "FIRST-NAME", 'A',20); Decl( 3, "NAME", 'A',20); Decl( 3, "MIDDLE-NAME", 'A' 20); Decl( 2, "SEX", 'A',1); Decl( 2, "FULL-ADDRESS"); Decl( 3, "ADDRESS-LINE", 'A',20,1,4); Decl( 3, "CITY", 'A',20); Decl( 3, "POSTAL-CODE", 'A',10); Decl( 3, "COUNTRY", 'A',3); Decl(1, "#NAME", 'A',40); //End-Define                     </pre>
<pre> 260 READ EMPLOYEES PHYSICAL 270 REJECT IF CTY NE 'PERPIGNAN' 280 WRITE 290 IT NAME 30T PERSONNEL-ID 300 END-READ                     </pre> <p>Business Logic Unchanged</p>	<pre> ReadPhysical ("EMPLOYEES"); { while (!EOF("EMPLOYEES"))   if (Var("CITY").ne("PERPIGNAN"))     continue;   Write (1,Var("NAME"));   Write (9, Var("PERSONNEL-ID"));   WriteRow (1);   Next ("EMPLOYEES");} Close ("EMPLOYEES");                     </pre>
<pre> 900 FIND (5) EMPLOYEES WITH PERSONNEL-ID &gt;20 950 REJECT IF SEX = 'M' 970 ACCEPT IF COUNTRY = 'UK' 980 COMPRESS FIRST-NAME NAME INTO #NAME 010 END-FIND                     </pre> <p>Easy to Maintain!</p>	<pre> Find (5,"EMPLOYEES", "PERSONNEL_ID &gt; 20"); while (!EOF("EMPLOYEES")) { if (Var("SEX").eq("M"))   continue;   if (Var("COUNTRY").eq("UK"))     continue;   Compress(false);   Comp (Var ("FIRST-NAME"));   Comp (Var ("NAME"));   Intro (Var ("#NAME"));   Next ("EMPLOYEES"); } Close ("EMPLOYEES");                     </pre>

The Local Data Area is translated into Java. The result is very similar to the Natural data definition. Note: dEMPLOYEES is an extension of the NatDDM base class.

Once declared as an instance of dEMPLOYEES, all database actions trigger the reading of corresponding tables and transfer of the contents to the defined data area. Updates will read the contents of the defined data area and will apply changes to the appropriate tables of the relational schema.

The Java method corresponding to 'Find' will accept a number of optional parameters which translate clauses such as WITH, WHERE, SORTED BY, (LIMIT) etc.

The Natural COMPRESS statement has many options. These are handled by the parameters passed to the Java Method "COMPRESS".

- Generation of control files, based on the new RDBMS schema, that can be used to create SQL load files for bulk loading of the Adabas data into the production RDBMS tables

The data migration process can be repeated as many times as needed. Data migration will almost always require some degree of data validation and mapping. Natural to Java can provide automated support for activities such as:

- Validation that a field specified as a date in Adabas and stored as an A6 actually contains valid YYYYMMDD occurrences and uses lo-values or blanks as a null value. Validation of implied foreign key existence and constraint checks
- Mapping an Adabas A6 date field to an RDBMS date field with blank occurrences replaced with NULL
- Mapping an Adabas SSN representation to an appropriate RDBMS representation
- Expansion or replacement of state, region or status codes

#### Performance tuning and customization

All application platforms and databases have specific issues when it comes to performance and customization. Particular performance issues that appear during testing can be tuned using a number of methods:

##### *Database Performance*

DBAs can tune the new RDBMS as required using techniques such as:

- Adding or removing database indexes
- Changing the functionality of a particular Natural to Java I/O method for better performance
- Adding new Natural to Java I/O methods to handle specific database access situations
- Replacing Natural to Java methods with native Java JDBC calls where needed

##### *Application Performance*

With the object-oriented nature of Java, cosmetic and functional changes can be made to many of the Natural to Java classes to customize the application where needed. Clerity provides you with a complete set of source code to allow for in-house customization or can contract for this service on an as required basis. Customization methods might include:

- Changing or adding Natural to Java methods to improve logic
- Replacing old Natural-like business logic with new object oriented business rule concepts
- Recoding sections in native Java statements

#### Complete end-to-end solutions

Clerity's deep mainframe and open systems service bench can provide project management for the complete lifecycle of any migration. Our service experts can tailor a Natural to Java migration solution with the level of service required to make your implementation successful, and our infrastructure experts can offer consultative advice for other necessary hardware and software components as required.

The combination of strong products, deep expertise, and flexible service offerings makes Clerity the best choice to migrate Natural code to a more cost-effective environment that encourages new business opportunities.

Find more details about this solution and others at [www.clerity.com](http://www.clerity.com).

### About Clerity

Clerity is a leading full-service provider of mainframe migration, modernization, and optimization solutions. Drawing from over 16 years of experience, Clerity recognizes that companies have significant investments in core applications and procedures and provides a wealth of low risk, high value tools, technology, and services to reduce IT costs without sacrificing current functionality and service level agreements. Headquartered in Chicago, Illinois with offices worldwide, Clerity has customers in all in major countries, including some of the largest financial services ISVs and Fortune-class end users.

Learn how Clerity can provide an evolutionary path forward for your application and data environments at [www.clerity.com](http://www.clerity.com).



9930 Derby Lane, Suite 202 • Westchester, IL 60154  
Phone 1-888-2-REHOST (or 1-630-981-6100)

© 2010 Clerity Solutions, Inc. All rights reserved. Clerity and UniKix are trademarks, or registered trademarks, of Clerity Solutions, Inc. in the United States and other countries.

All other marks are the property of their respective owners.