Secure Modernization of IT Applications

Industrialized Transformation of Software and Data

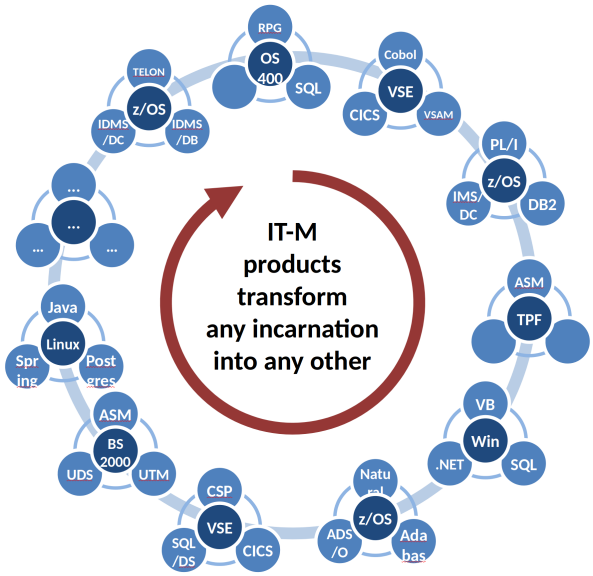
The pressure to modernize IT application systems is huge: It is caused by unbearable cost, lack of staff with ancient programming skills, architectural differences inhibiting integration with modern systems, and data kept in outdated formats and systems.

*ITM developed their **Njema product suite**, which allows carrying out secure, plannable, and affordable modernization projects, covering all aspects of individual IT application systems.*

Rationale

You can change the technology, but your application logic stays untouched!

An existing IT system is just one of many incarnations of the underlying business logic – you can change the DB system, the programming language, the operating system etc. - and still achieve the same functionality.



Njema fully complies to this philosophy.

Analysis – first things first!

ITM added efficient analysis capabilities to Njema in order to produce a comprehensive view of the application system quickly.

After the code – including Job Control, scheduler files or 3rd party product parameters, e.g. SAP – has been fed into Njema, its meta data repository knows exactly what is used, missing or obsolete, and how objects are connected with each other.

The analysis also reveals issues which might cause problems later during the project.

This allows ITM experts to quickly judge duration, effort, and cost of the intended modernization project.

Supported Products

Through the years, ITM has constantly extended the range of supported components. Currently we are supporting:

- Origin languages including PL/I, Cobol, CA-Gen, REXX, NatStar, RPG, Informix, Telon, Easytrieve, Natural, Pacbase, Uniface
- MVS, VSE, TPF, BS2000, and GCOS8
- Target languages: Java, C++, C#, Cobol
- Linux as the most popular target operating system. Of course migrations within the current O/S or to Windows, too.
- CICS, IMS/DC, TP8, UTM
- Mainframe Assembler to Cobol[1]
- MVS or VSE Job Control to Bash or another script language
- Schedulers TWS, CA-Schedule, TOM, JobTrac, HS5000 e.a.
- Oracle, DB2, MySQL, IMS/DB, PostgreSQL, MSSQL, Informix, Sesam, UDS, Adabas etc.
- All classic file systems
- Target frameworks like Spring, WildFly, TomCat, JSP and many more

Njema was developed in a way that new languages, data bases, GUI frameworks, and customer specific needs can be implemented efficiently.

Code Quality

Njema produces clean and easy-to-read target code: re-engineering and elimination of dead code or obsolete objects are just some of the available features.

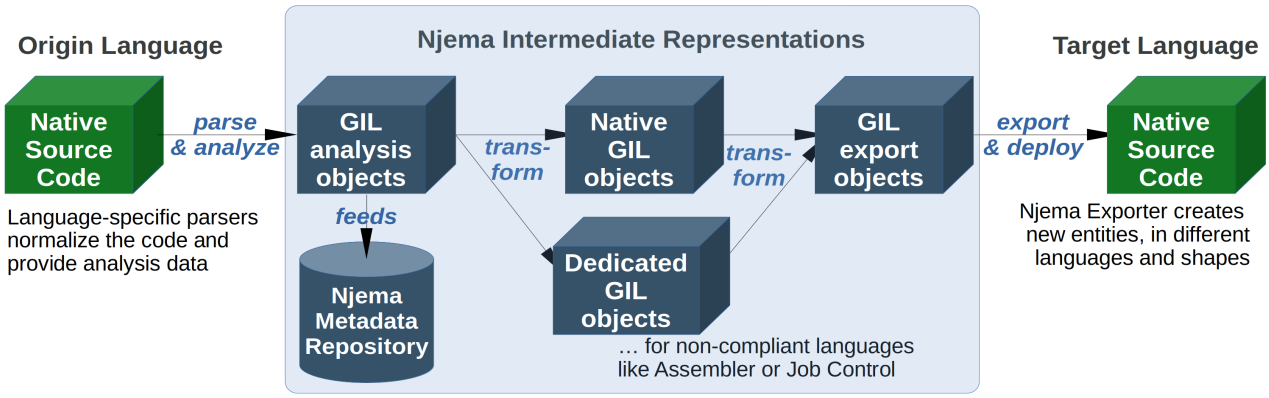
There are no hidden 3rd party products: Everything is delivered as source code.

Due to automation, processing rules are easily adapted. Afterwards, all programs can be transformed again without any human intervention. Customer standards are easy to integrate into the standard transformation rule sets.

1 A transformation to Java & Co may follow directly or later.

Language Transformation

Unlike typical “*converters*”, Njema starts transformation only after it analyzed all language objects filling the *Njema Metadata Repository*. This repository controls the transformation of source code and data.



As it is rule-based, this methodology offers an enormous flexibility. Njema contains a huge collection of rules for adapting the code related to language, operating system, TP environment, target framework and so forth.

So far, Njema has processed more than 300 million code lines.

Applying Njema in Projects

Depending on the concrete project or customer situation there are different ways to set up the Modernization Project as such. Here are some general scenarios:

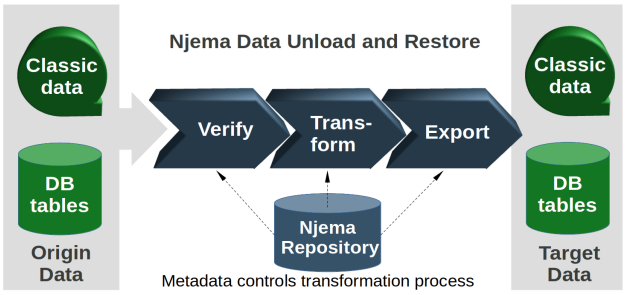
- **Packaged**
  The classic approach.
  Code and data is divided in packages, which are transformed and tested independently.
- **Cross-Compile**
  Code maintained in the old language.
  Njema acts like a compiler transforming the code to another language and environment, in which the code is executed productively.
- **Stepwise Transition**
  Like *Cross-Compile,* above, but a new development team takes over maintenance for transformed packages, one by one. This allows for a slow and controlled phase-out of the old technology
- **Big Bang**
  Another classic modernization approach.
  Often required, because of heavily interwoven code and data and/or cut-over time constraints.

Each of these scenarios has its pros and cons. The project-specific variation will be selected based on the analysis results, the future IT architecture, time constraints, and many other customer-related aspects.

Data Transformation

After the analysis, Njema knows all about data structures, schemes, views, files and fields. Based on this, it is able to move data from classic files to in-memory files or into a database. It can exchange the database system and much more to match future requirements.

This includes changes to field names, data formats and field content.



Terms and Conditions

A cost and effort estimate and a PoC transformation is available free of charge once the source code is analyzed and the future IT architecture is identified.

Transformation is provided as a service, or individual Njema Modernization “*Factories*” are installed in a cloud and licensed as a continuous-build package.