# Legacy Modernization

*The low-code platforms OutSystems and Mendix are frequently used to develop mobile apps and front-end web applications. Moreover, some organizations see the potential of low-code platforms to replace their legacy systems by rebuilding them. In this article we describe briefly the benefits of low-code, the business case, and the approach to rebuild a legacy application.*

**The benefits of a low-code platform**

OutSystems and Mendix provides a complete development and deployment environment in the cloud (platform as a service). They cover the development, testing and production environments. They automate error-free deployment on multi-devices: Android, iOS, Amazon, Azure. Through SOAP web services and REST API's they deliver an integration platform with back-end systems. Mobile, web and core (back-end) applications built with OutSystems or Mendix are enterprise grade: performant, scalable and secure.

OutSystems and Mendix speed up application and innovation delivery. They deliver a model-based, visual development and enable to run projects at digital clock speed, and continuously. They support the transition of application delivery methods into agile development (Scrum, DevOps). OutSystems and Mendix increase the development capacity by empowering citizen developers.

**The business case**

A positive business for rebuilding a legacy system is based on the very high productivity of the software developers using OutSystems or Mendix. They are characterized by a high level of automation. This implies that developers can concentrate on the business functionality and modelling instead of bothering with all kind of underlying, technical concerns.

Dependent of the nature and complexity of the application, we have experienced a productivity gain of 5 to 10 times compared to traditional programming languages, like Java and C#. A report of KPMG confirms: 30 – 50% more productive during development, 75% in deployment, and 50 – 75% during maintenance. The business case becomes more attractive when the infrastructure can be decommissioned, e.g., phasing out the mainframe or AS400.

**Package implementation versus bespoke development**

One of the appealing cases is an organization that wants to replace their legacy system. A standard package was found that would cover 70% of the desired functionality. That means that 30% should be customized and developed in the technology of the package. After an evaluation, this organization decided to rebuild the entire legacy system with OutSystems. Finally, 100% of the requirements were realized against 50% of the budget that was reserved for the package implementation.

**References**

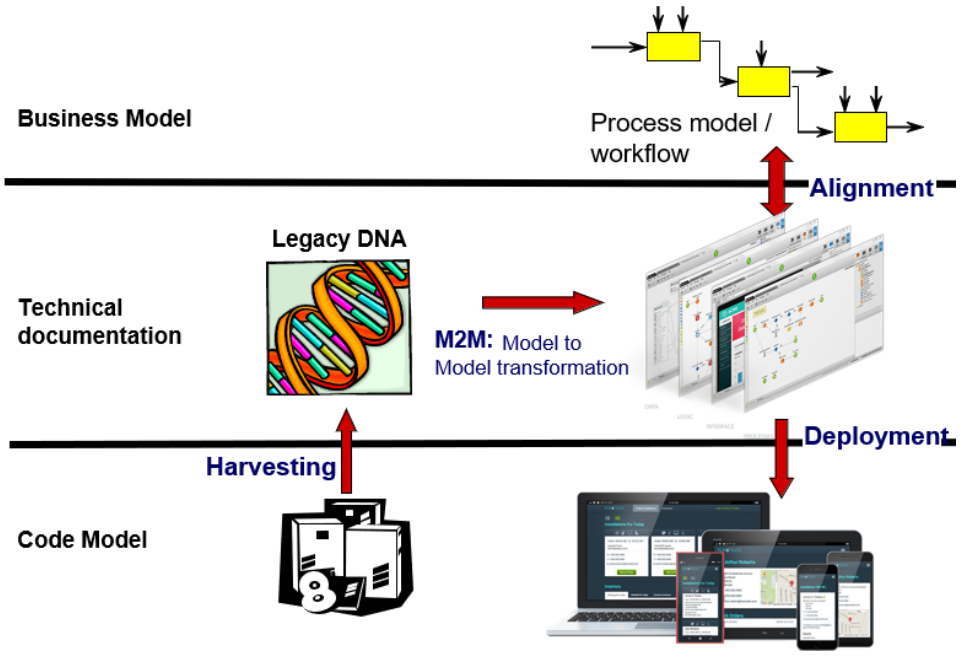
Atos has rebuilt several legacy systems with low-code, among them:

- Asset management and planning system, formerly a Java-based system. It is a highly sophisticated asset management system that four of the world's navies rely on for everything from tracking vessels, aircraft and munitions to scheduling maintenance and managing personnel. With more than 5,000 users in 120 locations, it's a serious, mission-critical application. Atos used OutSystems to deliver the needs of a modern navy. It is now deployed across 4 navies with 5000 + users and 120 on premise installations.

- Banking application, formerly a C#.NET front-end application. It is the online environment of a Dutch bank for its external customers: professional financial entities like institutional investors (pension funds and insurance companies). It provides reporting tools about positions and statuses, and their customers can execute transactions. Atos has modernized the application using OutSystems to improve overall user experience. Therefore, Atos has installed OutSystems on premise in that way that it meets the compliance and security policies of the bank.

**Approach**

Atos has a dedicated approach to rebuild legacy applications. It starts with extracting the DNA of the legacy application using an application mining tool. It scans the code and produces the technical documentation, e.g., the entity-relationship diagrams, high-level architecture view, database access view, and transaction/workflow view. Programming languages covered are Cobol, PL/SQL, JCL, JSP, HTML, ABAP, Oracle*Forms, PHP, Pro C, C, C++, C#, Tibco, VB Script, VB.NET, Visual Basic, XML, Java, JavaScript, among others. The low-code developers analyze the produced technical documentation and transform it into the low-code models. Based on these models a first version is deployed. Next, during a number of sprints, they enrich this version with business logic, work flow, authentication and integration with other systems. Each sprint ends with the deployment of new functionality, which is made available to the user community for acceptance test.




**About the author**

Kees Kranenburg is Solution lead Low-code platforms at Atos The Netherlands.
https://atos.net/en/expert/kees-kranenburg