M. Lytvynov, O. Lytvynov, V. Khandetsky, O. Hurko

ON MIGRATION OF LEGACY HOSPITAL INFORMATION SYSTEM TO DOMAIN-DRIVEN ARCHITECTURE

Year-to-year the complexity of software keeps growing. Today's business requires more than just an information system that responds the immediate needs. It requires more flexible, maintainable, and testable IT infrastructure able to meet the business challenges. Multi-layered and service-oriented [2] software architectures, Domain-driven Design (DDD) approach with its modifications [3], event-driven architecture can be seen as variants to handle the challenges. But the implementing of such approaches results in dozens of huge projects with hundreds and even thousands of classes and sophisticated testing infrastructure necessary to support continuous refactoring of the system. It complicates the development process which results in time and effort expenses required to realize, test, and implement the system into organization.

Modernizing legacy systems is one of the most challenging problems for developers often face when engineering information systems. Thus, changing requirements of the system environment, emerging new technologies and business models re-engineering, on one hand, and functionality, performance, usability, maintainability issues of the legacy systems, on the other, force the companies to start the migration process. All these statements can be fully attributed to modern hospital information systems. Health information systems (HIS) represent an essential part of the infrastructure for the delivery of good health care. And at this point the question of choosing the strategy of migration arises.

Forward or database first migration provides to migrate unchanged legacy applications forward onto a modern DBMS and then migrates the applications. The profit of the approach seemed to be achieving better data operations processing and data analysis performance, safety and relevancy using modern DBMS facilities.

While the database migration step could be made iteratively, it seems not very effective because it will complicate the cut-over as well as the gateway that would have to mediate between the diminishing legacy database and the growing target database.

This approach can only be applied to a legacy software where the Data-access layer is decoupled from the upper layer by the interfaces (in case when this layer ever exists in legacy software and separated from business logic layer).

The Reverse Migration or Database Last approach provides to migrate target applications in the reverse direction, back onto the legacy database until it is subsequently migrated. This method permits more time to deal with the database migration. It involves a reverse database gateway that facilitates a Chicken Little migration of the applications and their interfaces before the Big Bang database migration. The reverse database gateway converts all calls to the modern DBMS from target applications and maps them into calls to the legacy database service. It must also capture, translate, and direct responses from the legacy database service to the appropriate modules (means notifications).

Iteratively, it supports more target application modules until all are supported, thereby completely encapsulating the legacy database service. It contracts as the target applications are migrated to access the target database directly.

The advantages of the approach are as follows. Firstly, this approach can be used for semi-decomposable and even for non-decomposable legacy applications, while Forward Migration one cannot. Secondly, this approach is more commercially acceptable than the Forward Migration approach because it allows legacy application to operate uninterrupted while new application is being redeveloped.

The next important approach is the Iterative method. It implies that one component at a time is reengineered. Thus, the legacy system gradually evolves over a period.

The methodology enables simultaneous operations of the reengineered system and the components of the legacy system. The components of the reengineered system access either the legacy database or the new database, based upon the location of the actual data to be accessed.

Prototyping and piloting are recommended as the basic strategies to test potential solutions and validate system integrity, performance, and acceptance by users.

Prototypes should be meaningful and focused on evaluating user interfaces and operational usage scenarios. Pilot implementations on a small scale are suggested to validate migration efforts. Pilot approaches are particularly useful for gathering user input and achieving user acceptance.

Given the large volume of work on migrating legacy systems and improvement of their understanding, approaches are divided into two groups: modernization and replacement ones. Modernization, in turn, is classified based on two common strategies. On the one hand, there is the strategy of encapsulating inherited logic using a modern software layer, which is called wrapping, black boxing, or direct migration. On the other hand, there are «white box» strategies or indirect migrations that completely redefine the outdated system using reengineering principles.

In [1], the authors offer a «black box» method based on encapsulation, allowing interactive features of outdated systems to be made available as web services.

Thus, the migration approach should combine the properties of the Reverse and Iterative methods, using pilot deployment, as well as can work on the old database, create replicas, and connect new services to them (after implementing the pilots).

REFERENCES

1. Bisbal, J., Lawless, D., Wu, B. (1999). *Grimson J. Legacy information systems: issues and directions*, IEEE Software, 16 (5), 103–111.
2. Erl, T. (2016). *Service-Oriented Architecture: Concepts, Technology, and Design*. Pearson Education, Limited. 792 p.
3. Martin, R. C.(2017). *Clean Architecture: A Craftsman's Guide to Software Structure and Design.* Pearson Education Asia. 352 p.

K. Maksymenko, R. Bilichenko, N. Kaliberda

ADVANTAGES AND DISADVANTAGES OF ARTIFICIAL INTELLIGENCE

In recent years, artificial intelligence has been on everyone's lips, especially after the launch of such services as ChatGPT and Midjourney. Some admire the possibilities that this technology provides, while others, on the contrary, see neural networks as dangerous. So what are the advantages and disadvantages of artificial intelligence [2]?

Artificial intelligence (AI) is the intelligence of machines or software, as opposed to the intelligence of other living beings, primarily of humans. It is a field of study in computer science that develops and studies intelligent machines. Such machines may be called AIs.