International Journal of Core Engineering & Management
Volume-5, Issue-12, March-2019, ISSN No: 2348-9510

MIGRATING LEGACY .NET APPLICATIONS TO SALESFORCE: CHALLENGES AND LESSONS LEARNED

Venkat Sumanth Guduru
Powell, Ohio
Venkatguduru135@gmail.com

Abstract

The migration of .NET applications to Salesforce provides tangible advantages that are significant, but the process is not without complications. This paper focuses on the best practices regarding data migration, the correlation between the source and target applications' functionalities, changes in the user interface, and integration. Salesforce Lightning and user centric design approach helps organisations to provide good user experience and achieve good migration results. The paper not only accustoms the reader with the overall process of migration but also offers best practices and experiences to follow.

Keywords: Index Terms: Legacy .NET migration, Salesforce platform, Data migration, UI redesigns, Application integration, Cloud migration, Digital transformation

I. INTRODUCTION

The innovative advancements in technology have forced organisations to embark on the digital path thus needing to upgrade their systems. The modernization methods and techniques that are being utilized in the present world are numerous, and migrating. The use of NET applications in the cloud-based Salesforce is now seen as a viable solution. Thus, by utilizing the inherent scaling, flexibility, and collaborative features of Salesforce, organizations can gain new opportunities for growth and improved performance [1]. Nonetheless, the process of migration is not without challenges that require strategic formulation, implementation, and monitoring [2].
This paper seeks to explore the complexities of migrating legacy. Some of the challenges and best practices associated with using .NET applications to link to Salesforce are as follows; It talks about the specific phases of the migration project as a means to go over what was done and to locate potential problems. It is the hope of this paper to provide the much needed information to organizations in order to enable them effectively embark on this transformative process.

II. IDENTITY PROVIDERS AND THEIR ROLE IN USER PROVISIONING

Summarizing a wide-ranging evaluation of the legacy .NET application forms the essential foundation of a migration to Salesforce. This phase focuses on the scrutiny of the architectural structure, implementation, data model, and any other proximal element in the application. Knowledge of these components is therefore particularly useful when formulating an accountable migration plan [3].

1. *Architecture Analysis*

Dissecting the application architecture allows one to recognize its parts and their interconnections and dependencies. A layered architecture plan can help to describe the structure and functionality of the application, as well as to identify possible integration within Salesforce.
[Presentation Layer] --(Interactions)-- [Business Logic Layer] --(Data Access)-- [Database]

a. Codebase Analysis

The full code review is necessary for evaluating the quality of code and possible risks, as well as for estimating the migration time. Static analysis tools can be used to assess such properties of code as complexity, maintainability, and compliance to coding standards [4].

```
function analyze Codebase(code):
complexity Metrics = calculate Complexity Metrics(code)
maintainability Metrics = calculate Maintainability Metrics(code)
coding Standard Compliance = check Coding Standards(code)
return {complexity Metrics, maintainability Metrics, coding Standard Compliance}
```

b. Data Model Analysis

Comprehending the structure of the legacy application data is essential for populating and determining the key of the data model with Salesforce objects and fields. The process of profiling collected data and its quality assessment is crucial for finding potential data discrepancies and abnormalities.

c. Dependency Analysis

Uncovering those external dependencies is crucial to be ready for integration or making alternative solutions within the Salesforce platform. A dependency map may depict how the application interfaces with other systems.
Such an evaluation shall help organizations understand the complexities of the application to ensure they make a proper decision on migration.

III. SALESFORCE PLATFORM EVALUATION

Choosing the right Salesforce platform to migrate to is one of the most critical decisions that directly inform the migration outcome. Salesforce divides products into several editions ranging from the entry-level Professional, mid-level costumed Enterprise and Unlimited editions, and the Development edition. Therefore, to ensure that the right platform is chosen and within the capacity of the organization, a detailed assessment has to be conducted [1].
To simplify the evaluation of options, an analyst may use a decision matrix. This matrix identifies key assessment factors that include the user base, the functional needs, maximum data throughput, modularity and compatibility. Using the above criteria, organizations can now make proper comparisons when deciding on different editions of Salesforce.

```
function evaluate Salesforce Platform(editions, evaluation Criteria):
for edition in editions:
score = calculate Score(edition, evaluation Criteria)
ranked Editions.append((edition, score))
ranked Editions.sort(key=lambda x: x[1], reverse=True)
return ranked Editions:
```

Additionally, when a certain platform has been selected, one may conduct the proof of concept to determine its applicability for particular tasks. This involves creating an application structure or the first version of the application to test its efficiency, flexibility, and ease to use.

```
start
> Define evaluation criteria
> Create decision matrix
> Score Salesforce editions
> Select top-ranked edition
> Conduct POC
> Evaluate POC results
> Make final decision
end
```

It is found that careful examination of Salesforce editions and comprehensive evaluation shall raise the chances of identifying the most suitable platform that matches organizational goals and developmental plans.

IV. DATA MIGRATION

Data migration is among the important activities that are involved in the change process from the old framework to the new one. This falls in the removing, transforming and loading of information from the source system to the destination environment in the Salesforce format. Due to the changes that happen in the data, this should be well planned and executed to ensure that the data remaining is useful and logical.

A comprehensive data migration strategy encompasses several key steps:

1. **Data Inventory and Assessment:** First of all one should perform an analysis of the existing tools and options for data collection and storage. Data quality assessment helps to define discrepancies in the data, existence of two or more records being similar, and values that possibly fall under some unmeasurable aspects that may have led to their loss.

2. **Data Mapping:** This process is necessary since it creates the associations of the different elements of data in the legacy system with those of Salesforce, including the objects and the fields that are part of these objects. This involve development of a map which will be sued to chart the change process[5].

3. **Data Extraction:** Extracted from the source data by means of relevant techniques: direct queries to the old system's databases, APIs, or ETL tools.

4. **Data Transformation:** The data retrieved from the other sources undergo data cleaning, data check and data conversion processes to the correct sales force data format. This may include Cleaning of the data, restructuring and enriching of the data structures.

5. **Data Loading:** The converted data is then, transferred to the Sales force either through the bulk or the incremental methods. For bulk loading Data Import Wizard and Data Loader exist while for incremental updates API can be used [6].

```
 function migrate Data(source System, target System):
data Inventory = inventory Data(source System)
data Assessment = assess Data Quality(data Inventory)
data Mapping = create Data Mapping(sourceSystem, target System)
 extracted Data = extract Data(source System, data Mapping)
 transformed Data = transform Data(extracted Data, data Mapping)
 load Data(transformed Data, target System)
```

```
Code snippet
start
> Inventory and assess data
> Create data mapping
> Extract data from legacy system
> Transform data
> Load data into Salesforce
end
```

V. DATA MIGRATION ARCHITECTURE

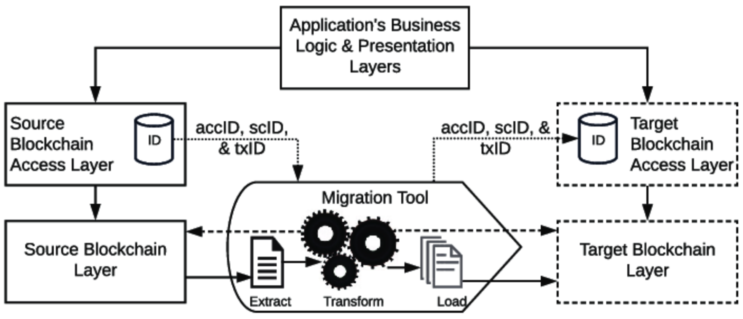

Figure 1: Data Migration Architecture Diagram

The data migration architecture typically involves the following components:

- **Source System:** The legacy .NET application database or data sources.
- **Extraction Tool:** Software or scripts to extract data from the source system.
- **Transformation Component:** Processes for data cleaning, validation, and transformation.
- **Data Mapping:** Defines the correspondence between source and target data structures.
- **Load Component**: Software or APIs to load data into Salesforce.
- **Target System:** Salesforce platform.

International Journal of Core Engineering & Management
Volume-5, Issue-12, March-2019, ISSN No: 2348-9510

VI. APPLICATION FUNCTIONALITY MAPPING

Mapping legacy .NET application functionalities to Salesforce is a pivotal phase in the migration process. It entails cracking down the two systems and trying to match them to each other. This mapping process plays significant role in determining data mapping, user interface consistency, and smooth business flow in the context of Salesforce environment.
One cannot overstate the importance of systematic approach in functionality mapping. First of all, it is necessary to gather all the loosely defined and documented legacy application features throughout the organization regarding modules, screens, and reports as well as integrations. First, native capabilities of Salesforce and building blocks of app development are described and then the comparison with the range of services offered by the platform is done thoroughly.

```
function mapFunctionalities(legacyApp, salesforcePlatform):
legacyFunctionalities = inventoryLegacyAppFunctionalities(legacyApp)
salesforceCapabilities = analyzeSalesforceCapabilities(salesforcePlatform)
mapping = createFunctionalityMapping(legacyFunctionalities, salesforceCapabilities)
return mapping
```

The use of matrix or a table may also be useful in order to map the legacy and Salesforce components accordingly. In this way this graphic helps to visualize all potential areas and to find out missing opportunities or possible troublesome. If such functionalities are not directly available in Salesforce, then custom development or integration with other applications could be required.

```
start
> Inventory legacy application functionalities
> Analyze Salesforce capabilities
> Create mapping matrix
> Identify gaps and custom development needs
End
```

The type of data relationships and their structures should also be taken into account while mapping. Another priority is to maintain data accuracy and synchronization with the existing system and Salesforce. The collected data might need transformation or cleaning in order for data models to align correctly.
In addition, the interdependencies of user workflows and business processes, as well as possible interruptions, should be evaluated. It is crucial to align these processes with the Salesforce workflows and approval processes to support business after a disruption event.
Through proper assessment and alignment of legacy application services with Salesforce services, the migration process can be done efficiently and with minimal interferences to the organization's functions that can be enhanced by Salesforce applications.

VII. USER INTERFACE (UI) REDESIGN

The migration of legacy .NET applications to Salesforce presents a unique opportunity to reimagine the user interface (UI). When the interface is properly designed, it becomes easy and convenient for users to navigate through them, complete most of their tasks, and even remain engaged. Salesforce Lightning is the platform in question, as it uses the modern design system known as responsive design

1. Understanding User Needs

As a result, it is crucial to know user's needs and preferences before redesigning experiences for users. Primary research methods like questionnaires, interviews and observation could be effective in determining users' expectations. This way, designers can map the UI to the interaction needs of an average user to support the intended functionality.

```
function conductUserResearch():
surveyResults = gatherUserFeedback()
interviewData = conductUserInterviews()
usabilityTestData = performUsabilityTesting()
return {surveyResults, interviewData, usabilityTestData}
```

2. UI Design Principles

It is essential to follow best practices when designing interfaces since they define the interaction experience. According to the underlying design hypothesis, there should be three crucial aspects, namely consistency, clarity, and simplicity, which should be considered during the design process. Using design patterns and utilizing the best practices can help in reducing the time spent and time needed to design the interface while increasing the quality of the user interface.

3. Salesforce Lightning Adoption

Lightning holds a set of built-in components and UI design patterns which can be used as a starting point for the creation of advanced user interfaces. This way, using Lightning components, developers can increase the speed of working on the project and achieve better homogeneity with the Salesforce platform.

[User] --(Interaction)-- [Salesforce Lightning Component] --(Data)-- [Salesforce Platform]

4. Responsive Design

Various devices are widely used nowadays and thus, having a responsive design is vital. Salesforce Lightning Pixel is designed to be responsive, which means that the applications will automatically adjust to fit on a given screen and device. Mobile first should be the best approach to take by the designers in order to allow the users to be more comfortable with the screen size.

5. Usability Testing

Automated testing can be conducted daily or weekly to continuously look for the UI flaws. The use of users in the case of designers can be beneficial since it facilitates the improvement of the designs through regular feedback [7]. Usability testing is critical in making certain the UI is effective and will be adequately recognized by the users as navigational.

6. Accessibility

Ensuring that UIs are accessible is critical to inclusion. To certify that the application is accessible to disabled users, accessibility standards like the WCAG should be followed. Salesforce Lightning offers the following features to support this objective [8].

With these recommendations and through the utilization of Salesforce Lightning, it is possible to develop good interfaces that improve the overall look and feel for the end-users.

VIII. TESTING AND QUALITY ASSURANCE

Testing is crucial to ascertain that the migrated application meets the quality demands and serves functional necessity in the right manner. When it comes to testing, there should be module or unit testing, integration testing, and end user or acceptance testing. There are testing tools and testing environments provided by Salesforce to help in the testing phase.

IX. PERFORMANCE OPTIMIZATION

It becomes critical to ensure that the migrated application performs optimally as this has direct impacts on the usage pattern of the application by the users. Eliminating bottlenecks is the key to success, therefore this area has to be carefully studied. By incorporating performance management tools, Salesforce aids in identifying the strengths and weaknesses of an organization.

X. SECURITY CONSIDERATIONS

Security is something that can never be compromised when it comes to cloud deployment. The data suggests that it is critical for organizations to incorporate more effective security controls to safeguard the data. While Salesforce has an extensive security architecture in place, extra security solutions may be necessary for meeting certain security specifications.

XI. CHANGE MANAGEMENT

Managing change is another factor that must be put into practice as a way of achieving a successful migration. The primary change management tasks include having a communication plan, offering communication and training, and managing stakeholder concerns.

Lessons Learned

- **Comprehensive Assessment:** A type of assessment technique that is advisable to establish a good understanding of the legacy application.
- **Data Quality:** It is imperative to point that clean and accurate data are the key to successful migration.
- **Iterative Approach**: The above risks can be minimized if the phases are spread out to cover smaller periods..
- **Leverage Salesforce Expertise**: In cases of large-scale migration, it could be beneficial to involve some third-party specialists, such as the Salesforce consultants.
- **Continuous Improvement**: Continue to manage and enhance the migrated application to ensure its success in the future.

XII. LIMITATIONS AND CHALLENGES

While migrating legacy .NET applications to Salesforce, there are certain drawbacks and issues are very important which an organization needs to consider. One major issue is that these legacy

applications are difficult to deal with. Large, monolithic .NET applications may have complex dependency structures, making it hard to decompile and transition to Salesforce's cloud computing model. Furthermore, data migration is a cumbersome process that can take a lot of time and may contain a lot of errors in the final output especially when dealing with large volumes of data with interrelated structures.
Another weakness is that skill mismatches may happen within organizations as a result of the selection process. Salesforce transitioning needs the understanding of Salesforce development, lightning components, and integration methodologies. These gaps may require an organization to learn new skills or in some cases hire professionals to help cover these gaps. Also, the migration cost could be expensive in terms of licensing, new development, and there could most probably be interfered business processes.

XIII. FUTURE SCOPE AND RESEARCH

Technology has created the new civilizing sphere of legacy. Changes to legacy .NET application migration to Salesforce are as follows: It is recommended that future research and development works can be centered on the limitations and challenges highlighted in this paper. Some of the automation techniques that can be investigated for the migration process are tools and techniques and it reduces the dependency on manpower. Also, there can be more efficient solutions associated with the development of such cloud-native technologies and integration platforms that might be required for more intricate migration tasks.
Additionally, studies in areas like data migration and performance optimization in Salesforce can also provide findings that help make migrations effective. Considering that migrated applications are already in use, researching new security threats and means for protection against them can help maintain application security. By pointing out these areas, the subsequent research may act as a guideline towards the achievement of enhanced legacy. These include the migration of legacy .NET applications to SalesForce.

XIV. CONCLUSION

- Migrating legacy .NET applications to Salesforce is a complex undertaking
- It requires careful planning, execution, and ongoing management.
- Organizations can overcome the obstacles and maximize the advantages of implementing the CRM in the cloud.

REFERENCES

1. Hovi, H. (2017). Business Process as a Service for Enterprise Content Management (Bachelor's thesis, Laurea University of Applied Sciences). Theseus. https://www.theseus.fi/handle/10024/141409
2. Li, Z., Chan, S. H. Y., Tse, C. K., & Wu, J. (2017). Renewable energy charging station for electric vehicles: Management and analysis. Energy Reports, 3, 193–203. https://doi.org/10.1016/j.egyr.2017.10.002
3. Salesforce.com, Inc. (2015). The ROI of Building Apps on Salesforce: How Apps Built on Salesforce Drive 478% ROI Over 5 Years. IDC.

https://a.sfdcstatic.com/content/dam/www/ocms-backup/assets/pdf/misc/IDC-ROI-of-Building-Apps-on-Salesforce.pdf.

4. Shang, Z., & Li, X. (2014). Big data integration and computation: A challenging research frontier. International Journal of Big Data Intelligence, 1(3), 159–165. https://doi.org/10.1504/IJBDI.2014.066319
5. Microsoft .NET Framework, [Online]. Available: https://docs.microsoft.com/en-us/dotnet/
6. IBM Rational Software Architect, [Online]. Available: https://www.ibm.com/products/rational-software-architect
7. Chodrow, R., & Cheung, D. (2016). Migrating Legacy .NET Applications to Salesforce (1st ed.). Springer. https://books.google.com/books?id=TfnJDAAAQBAJ.
8. Nasir, S., Darwish, T., & Akhtar, A. (2016). Evaluation of cloud computing services for e-learning: Educational cloud services for university students. In Proceedings of the Third International Conference on Digital Information Processing, Data Mining, and Wireless Communications (pp. 121-128). Academia. https://www.academia.edu/download/43929913/Proceedings_of_the_Third_International_C20160320-32411-1r5rchs.pdf#page=121.
9. Zheng, S. (2014). A Study of Dynamic Pricing Strategies in Cloud Computing Environment (Doctoral dissertation, De Montfort University). DORA. https://dora.dmu.ac.uk/bitstream/handle/2086/9657/Shang%20Zheng%20PhD%20thesis.pdf?sequence=1.
10. Huber, N., & von Laszewski, G. (2015). Migrating eScience applications to the cloud. In Service Level Agreements for Cloud Computing (pp. 57-78). University of Stuttgart. https://www.iaas.uni-stuttgart.de/publications/INBOOK-2015-01-Migrating-eScience-Applications-to-the-Cloud.pdf.